UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

HUGO HISSASHI MIYATA

**Contribuições para o problema *flow shop* com bloqueio, tempos de *setup* dependentes da
sequência e funções-objetivo hierárquicas sujeitas ao custo total de manutenção
preventiva**

SÃO CARLOS
2020

HUGO HISSASHI MIYATA

Contribuições para o problema *flow shop* com bloqueio, tempos de *setup* dependentes da sequência e funções-objetivo hierárquicas sujeitas ao custo total de manutenção preventiva

Tese de doutorado apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo como parte dos requisitos para obtenção do título de doutor em Engenharia de Produção.

Área de Concentração: Gestão de Operações

Orientador: Prof. Dr. Marcelo Seido Nagano

SÃO CARLOS

2020

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da
EESC/USP com os dados inseridos pelo(a) autor(a).

MM 618 Mi	<p>Miyata, Hugo Hissashi</p> <p>Contribuições para o problema flow shop com bloqueio, tempos de setup dependentes da sequência e funções hierárquicas sujeitas ao custo total de manutenção preventiva / Hugo Hissashi Miyata; orientador Marcelo Seido Nagano. São Carlos, 2019.</p> <p>Tese (Doutorado) - Programa de Pós-Graduação em Engenharia de Produção e Área de Concentração em Processos e Gestão de Operações -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.</p> <p>1. programação de operações. 2. flow shop. 3. bloqueio. 4. setup dependente da sequência. 5. manutenção preventiva. I. Título.</p>
--------------	---

FOLHA DE JULGAMENTO

Candidato: Engenheiro **HUGO HISSASHI MIYATA**.

Título da tese: "Contribuições para o problema *flow shop* com bloqueio, tempos de *setup* dependentes da sequência e funções-objetivo hierárquicas sujeitas ao custo total de manutenção preventiva".

Data da defesa: 17/01/2020.

Comissão Julgadora:

Resultado:

Prof. Dr. **Marcelo Seido Nagano**
(Orientador)

(Escola de Engenharia de São Carlos/EESC)

Aprovado

Prof. Dr. **Roberto Fernandes Tavares Neto**

(Universidade Federal de São Carlos/UFSCar)

APROVADO

Prof. Dr. **Leandro Callegari Coelho**

(Université Laval)

Aprovado

Profa. Associada **Debora Pretti Ronconi**

(Escola Politécnica/EP-USP)

Aprovado

Prof. Associado **José Francisco Ferreira Ribeiro**

(Faculdade de Economia, Administração e Contabilidade/FEA-USP)

APROVADO

JFR

Coordenador do Programa de Pós-Graduação em Engenharia de Produção:

Prof. Dr. **Marcelo Seido Nagano**

Presidente da Comissão de Pós-Graduação:

Prof. Titular **Murilo Araujo Romero**

Resumo

MIYATA, Hugo H. **Contribuições para o problema *flow shop* com bloqueio, tempos de *setup* dependentes da sequência e funções-objetivo hierárquicas sujeitas ao custo total de manutenção preventiva.** 2020. 265 p. Tese (Doutorado em Engenharia de Produção) – Departamento de Engenharia de Produção, Universidade de São Paulo, São Carlos, 2020.

O problema de programação de operações ou *scheduling* é um processo de decisão que desempenha um papel crucial em muitas indústrias de manufatura e de serviços. Sua essência concentra-se nas atividades de alocação e sequenciamento. Neste contexto, é de grande importância o desenvolvimento de técnicas que auxiliem na tomada de decisão da sequência de produção. De acordo com a revisão de literatura realizada nesta pesquisa, análises e proposição de técnicas de solução para o denominado ambiente de produção *flow shop* sem estoques intermediários entre as máquinas (bloqueio) vem ganhando notoriedade nas ultimas décadas, na sua maioria, sem a adição de restrições adicionais. Levando-se em conta a lacuna encontrada na literatura para esta área, nesta tese são propostos métodos de solução para o problema de programação de operações em ambientes *flow shop* com m máquinas, considerando bloqueio, tempos de *setup* dependentes da sequência e ocorrência de manutenção preventiva. As funções-objetivo tratadas foram a minimização da duração total da programação e do tempo total de fluxo dado que o custo total de manutenção preventiva exigido pela sequência não ultrapasse um dado valor. Um modelo de programação linear inteira mista (MILP) e formas de cálculo das datas de início e término das tarefas. Uma busca local para avaliação do reposicionamento das MP's foi desenvolvida para minimização da função-objetivo. Métodos heurísticos (construtivos e metaheurísticas) encontrados para os problemas de minimização da duração total da programação (C_{max}) e do tempo total de fluxo ($\sum_i C_i$) foram adaptados e propostos, no intuito de minimizar as funções-objetivo. Os métodos foram implementados e testados em dois cenários de ocorrência de manutenções preventivas, com instâncias de problemas de pequeno, médio e grande portes. Resultados estatísticos mostraram que em comparação com o MILP proposto, as metaheurísticas *Iterated Greedy* e GRASP(HPF2) obtiveram os melhores desempenhos para a minimização de C_{max} e de $\sum_i C_i$, respectivamente, em problemas de pequeno porte. Para problemas de médio e grande portes, *Iterated Greedy* e Variable Block Insertion Heuristic (VBIH) apresentaram o melhor desempenho geral nos dois cenários avaliados. Por fim, pode-se constatar experimentalmente que a busca local proposta obteve

melhorias significativas e expressivas na qualidade da solução gerada pelas metaheurísticas, o que sugere seu uso como parte da metodologia de solução dos problemas estudados.

Palavras-chave: programação de operações, *flow shop*, bloqueio, *setup* dependente da sequência, custos de manutenção preventiva.

Abstract

MIYATA, Hugo H. **Contributions for the m -machine blocking flow shop scheduling problem with sequence-dependent setup times and hierarchical objective-functions subject to preventive maintenance total cost.** 2020. Ph.D thesis, 265 p. (Ph.D in Production Engineering) – Production Engineering Department, São Paulo University, São Carlos, 2020.

Scheduling is a decision process that plays a crucial role in many manufacturing and service industries. Its essence focuses on allocation and sequencing decisions. In this context, it is of great importance the development of techniques that can be used as a support to the decision making of the production sequence. According to the literature review carried out in this research, analysis and proposition of solution techniques for the so-called blocking flow shop scheduling problem has gained notoriety over the last decades, mostly without the additional constraints. Taking into account the gap found in the literature for this area, in this thesis is addressed the m -machine blocking flow shop scheduling problem, with sequence-dependent setup times, preventive maintenance operations occurrences and minimization of hierarchical objective functions. The research aims to propose solution techniques to minimize the makespan (C_{max}) and the total flow time ($\sum_i C_i$) given that the preventive maintenance total cost required by the sequence does not exceed a given value. A mixed integer linear programming (MILP) and forms of objective-function computation were proposed. A local search for MP rescheduling was developed, aiming to minimize the objective-functions. Heuristic methods of the literature for the minimization of C_{max} and $\sum_i C_i$ have been adopted to minimize the preventive maintenance total cost and the hierarchical objective-functions of the approached problems. The methods were implemented and tested over two preventive maintenance instance problems configurations. Statistical results showed that, compared to the proposed MILP, the Iterated Greedy and GRASP(HPF2) obtained the best performances for the C_{max} and $\sum_i C_i$ minimization, respectively. For medium and large problems, Iterated Greedy and VBIH showed the best overall performances in both evaluated scenarios. Finally, it can be seen that the proposed local search significantly improved the quality solution generated by the metaheuristics, which suggests its use as part of the solution methodology of the problems studied here.

Keywords: scheduling; blocking flow shop; sequence-dependent setup times; preventive maintenance costs.

Dedico este trabalho aos meus pais Mieko Koura Miyata (*in memorian*) e Antônio Takashi Miyata (*in memorian*) e aos meus irmãos, que sempre incentivaram e apoiaram minhas decisões.

AGRADECIMENTOS

Aos meus pais Antônio Takashi Miyata (*in memorian*) e Mieko Koura Miyata (*in memorian*) e meus irmãos por todo suporte emocional, espiritual e financeiro durante todos esses anos.

Ao meu orientador Prof. Dr. Marcelo Seido Nagano, por todo auxílio e compartilhamento de sua experiência, conhecimento e conselhos. Suas valiosas orientações foram fundamentais para todo o processo de desenvolvimento desta tese.

À Escola de Engenharia de São Carlos (EESC-USP) e também ao Programa de Pós-Graduação em Engenharia de Produção (PPGEP/EESC-USP) por todo o suporte dado ao disponibilizar os recursos necessários para concretização desta pesquisa.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro por meio da bolsa de doutorado concedida.

Aos meus colegas e amigos do *Laboratory of Applied Operational Research* (LAOR) por todo auxílio e apoio.

LISTA DE FIGURAS

Figura 1 - <i>Flow Shop</i> permutacional com 4 tarefas e 4 máquinas.....	30
Figura 2 - Taxonomia dos métodos de solução	43
Figura 3 - Problema <i>flow shop</i> com duas máquinas, com bloqueio, com tempos de <i>setup</i> dependentes da sequência e com ocorrências de manutenção preventiva de dois componentes reparáveis.....	97
Figura 4 - <i>Flow shop</i> com 5 tarefas e com 4 máquinas	103
Figura 5 - Pseudocódigo do procedimento de posicionamento de MP's Prog_PM_Ruiz	105
Figura 6 - Pseudocódigo do procedimento de avaliação da posição das manutenções preventivas	107
Figura 7 - Pseudocódigo do procedimento de reposicionamento das atividades de manutenção preventiva	109
Figura 8 - Heurística do vizinho mais próximo para tempos de <i>setup</i>	118
Figura 9 - Pseudocódigo da heurística NEH	118
Figura 10 - Pseudocódigo do mecanismo de inserção NEH(λ)_Insertion_Scheme	119
Figura 11 - Pseudocódigo da heurística PF/PFE	120
Figura 12 - Pseudocódigo da heurística MM/MME.....	122
Figura 13 - Pseudocódigo da heurística wPF	123
Figura 14 - Pseudocódigo da heurística PW.....	125
Figura 15 - Pseudocódigo das heurísticas PF-NEH(x), wPF-NEH(x) e PW-NEH(x)	126
Figura 16 - Pseudocódigo da heurística PFT-NEH(x).....	129
Figura 17 - Pseudocódigo da heurística PFT-RZ(x)/PFT-RZ(x) _{st}	130
Figura 18 - Pseudocódigo da heurística PF+NEH(λ)/PF+NEH(λ) _{st}	131
Figura 19 - Pseudocódigo do algoritmo IG	131
Figura 20 - Pseudocódigo de IG_IJ e IG_RIS	132
Figura 21 - Pseudocódigo das busca locais: a) RIS e b) BestSwap.....	133
Figura 22 - Pseudocódigo de DWWO.....	135
Figura 23 - Pseudocódigo da fase de propagação	137
Figura 24 - Pseudocódigo da técnica Path Relinkg usada na fase de refração.....	138
Figura 25 - Pseudocódigo da Referenced Permutation Search	139
Figura 26 - Pseudocódigo geral do VNS proposto	140
Figura 27 - Calibração do parâmetro α da heurística MM nos cenários a) YS e b) RGM.....	141

Figura 28 - Calibração do parâmetro α da heurística MM-B nos cenários a) YS e b) RGM.	142
Figura 29 - Calibração do parâmetro λ da heurística mNEH nos cenários a) YS e b) RGM.	142
Figura 30 - Calibração do parâmetro λ da heurística mNEHst nos cenários a) YS e b) RGM	143
Figura 31 – Gráfico de médias do desvio relativo ao nível de 5% de significância de heurísticas construtivas para minimização do custo e C_{max} no cenário YS	146
Figura 32 – Gráfico de médias do desvio relativo ao nível de 5% de significância de heurísticas construtivas para minimização do custo e C_{max} no cenário RGM	146
Figura 33 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação ao número de tarefas no cenário YS	150
Figura 34 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação ao número de tarefas no cenário RGM	151
Figura 35 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação ao número de máquinas no cenário YS	151
Figura 36 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação ao número de máquinas no cenário RGM	152
Figura 37 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos tempos de <i>setup</i> no cenário YS	152
Figura 38 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos tempos de <i>setup</i> no cenário RGM	153
Figura 39 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos níveis de manutenção preventiva no cenário YS	153
Figura 40 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos níveis de manutenção preventiva no cenário RGM	154
Figura 41 - Desempenho das heurísticas do trade-off entre desvio relativo e tempo computacional no cenário YS	155
Figura 42 - Desempenho das heurísticas do trade-off entre desvio relativo e tempo computacional no cenário RGM	156
Figura 43 - Desempenho de IG em relação ao parâmetro d nos cenários: a) YS e b) RGM.	160
Figura 44 - Desempenho de IG em relação ao parâmetro τP nos cenários: a) YS e b) RGM	160
Figura 45 - Desempenho de IG em relação ao parâmetro jP nos cenários: a) YS e b) RGM	161
Figura 46 - Desempenho do algoritmo no cenário YS em relação aos parâmetros: a) população e b) h_{max}	162

Figura 47 - Desempenho do algoritmo no cenário YS em relação aos parâmetros: a) ω e b) λ	162
Figura 48 - Interação entre os parâmetros $\text{pop} * \omega$ no cenário YS	164
Figura 49 - Interação entre $\text{pop} * \lambda$ no cenário YS	165
Figura 50 - Interação entre $\omega * \lambda$ no cenário YS	165
Figura 51 - Desempenho do algoritmo no cenário RGM em relação aos parâmetros: a) população e b) $h_{\max}$	166
Figura 52 - Desempenho do algoritmo no cenário RGM em relação aos parâmetros: a) ω e b) λ	166
Figura 53 - Desempenho do algoritmo no cenário RGM para interação entre $\text{pop} * \lambda$	169
Figura 54- Gráfico de Médias ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte nos cenários a) YS e b) RGM	172
Figura 55 - Desempenho das metaheurísticas em relação à n para problemas de pequeno porte nos cenários a) YS e b) RGM	174
Figura 56 - Desempenho das metaheurísticas em relação à m para problemas de pequeno porte nos cenários a) YS e b) RGM	175
Figura 57 - Desempenho das metaheurísticas em relação ao <i>setup</i> para problemas de pequeno porte nos cenários a) YS e b) RGM	175
Figura 58 - Desempenho das metaheurísticas em relação à l para problemas de pequeno porte nos cenários a) YS e b) RGM	176
Figura 59 - Gráfico de médias do desvio relativo ao nível de 95% de confiança para problemas de médio e grande porte nos cenários: a) YS e b) RGM	180
Figura 60 - Desempenho das metaheurísticas em relação à n para problemas de médio e grande porte nos cenários: a) YS e b) RGM	180
Figura 61 - Desempenho das metaheurísticas em relação à m para problemas de médio e grande porte nos cenários: a) YS e b) RGM	181
Figura 62 - Desempenho das metaheurísticas em relação aos tempos de <i>setup</i> para problemas de médio e grande porte nos cenários: a) YS e b) RGM	182
Figura 63 - Desempenho das metaheurísticas em relação à l para problemas de médio e grande porte nos cenários: a) YS e b) RGM	182
Figura 64 - Pseudocódigo das heurísticas HPF1/NHPF1	187
Figura 65 - Pseudocódigo do procedimento para obtenção do índice $R(i)$	188
Figura 66 - Pseudocódigo das heurísticas HPF2/NHPF2	188
Figura 67 - Pseudocódigo de tPF	190

Figura 68 – Pseudocódigo das heurísticas tPF-NEH(x)	191
Figura 69 - Pseudocódigo de GRASP(HPF2)	192
Figura 70 – Pseudocódigo do algoritmo VNS	193
Figura 71 – Pseudocódigo do procedimento TNO	196
Figura 72 – Pseudocódigo da metaheurística DABC	197
Figura 73 – Pseudocódigo da metaheurística VBIH	198
Figura 74 – Pseudocódigo do procedimento de inserção bIM	199
Figura 75 – Pseudocódigo do algoritmo VLS	199
Figura 76 - Calibração do parâmetro μ da heurística HPF1 nos cenários a) YS e b) RGM...201	
Figura 77 - Calibração do parâmetro μ da heurística tPF nos cenários a) YS e b) RGM201	
Figura 78 - Calibração do parâmetro α da heurística MMA nos cenários a) YS e b) RGM ..201	
Figura 79 - Calibração do parâmetro α da heurística MMA nos cenários a) YS e b) RGM ..202	
Figura 80 - Gráfico de médias ao nível de 5% de significância das heurísticas para minimização do custo total de manutenção preventiva e tempo total do fluxo nos cenários: a) YS e b) RGM	204
Figura 81 - Desempenho das heurísticas construtivas em relação à n para problemas de médio e grande porte nos cenários: a) YS e b) RGM.....206	
Figura 82 - Desempenho das heurísticas construtivas em relação à m para problemas de médio e grande porte nos cenários: a) YS e b) RGM.....207	
Figura 83 - Desempenho das heurísticas construtivas em relação à duração do <i>setup</i> para problemas de médio e grande porte nos cenários: a) YS e b) RGM	207
Figura 84 - Desempenho das heurísticas construtivas em relação aos níveis de manutenção preventiva para problemas de médio e grande porte nos cenários: a) YS e b) RGM.....208	
Figura 85 - Desempenho das heurísticas construtivas em relação ao trade-off entre tempo computacional e desvio relativo no cenário YS	210
Figura 86 - Desempenho das heurísticas construtivas em relação ao trade-off entre tempo computacional e desvio relativo no cenário RGM	210
Figura 87 - Desvios relativos da interação entre n e ($\min\mu$; $\max\mu$) no cenário YS	213
Figura 88 - Desvios relativos da interação entre n e ($\min\mu$; $\max\mu$) no cenário YS	213
Figura 89 - Desvios relativos médios para os valores adotados para a População nos cenários: a) YS e b) RGM.....216	
Figura 90 - Desvios relativos médios para os valores adotados para a População nos cenários: a) YS e b) RGM.....216	

Figura 91 - Desvios relativos médios para os valores adotados para bS nos cenários: a) YS e b) RGM	218
Figura 92 - Desvios relativos médios para os valores adotados para $mSmax$ nos cenários: a) YS e b) RGM.....	219
Figura 93 - Desvios relativos médios para os valores adotados para τP nos cenários: a) YS e b) RGM	219
Figura 94 - Gráfico de médias das metaheurísticas ao nível de 5% de significância para problemas de pequeno porte nos cenários: a)YS e b) RGM.....	224
Figura 95 – Desempenho das metaheurísticas em relação ao número de tarefas para problemas de pequeno porte nos cenários: a)YS e b) RGM	224
Figura 96 – Desempenho das metaheurísticas em relação ao número de máquinas para problemas de pequeno porte nos cenários: a)YS e b) RGM.....	225
Figura 97 – Desempenho das metaheurísticas em relação à duração do Setup para problemas de pequeno porte nos cenários: a)YS e b) RGM	226
Figura 98 – Desempenho das metaheurísticas em relação aos níveis de manutenção preventiva para problemas de pequeno porte nos cenários: a)YS e b) RGM.....	226
Figura 99 - Gráfico de médias ao nível de 5% de significância das metaheurísticas em problemas de médio e grande porte nos cenários: a) YS e b) RGM	229
Figura 100 - Desvio relativo médio das metaheurísticas em relação à n para problemas de médio e grande porte nos cenários: a) YS e b) RGM.....	231
Figura 101 - Desvio relativo médio das metaheurísticas em relação à m para problemas de médio e grande porte nos cenários: a) YS e b) RGM.....	231
Figura 102 - Desvio relativo médio das metaheurísticas em relação ao $setup$ para problemas de médio e grande porte nos cenários: a) YS e b) RGM.....	232
Figura 103 - Desvio relativo médio das metaheurísticas em relação ao nível de manutenção para problemas de médio e grande porte nos cenários: a) YS e b) RGM	233

LISTA DE TABELAS

Tabela 1 - Resultado das buscas nas bases de dados para problemas <i>flow shop</i> com bloqueio	45
Tabela 2 - Publicações encontradas para o problema flow shop com minimização da C_{max} ...	46
Tabela 3 - Publicações que adotaram funções baseadas no tempo de fluxo	72
Tabela 4 - Publicações que consideraram funções baseadas em data de entrega.....	77
Tabela 5 - Publicações que consideraram funções baseadas em tempo de ciclo	80
Tabela 6 - Publicações encontradas que adotaram função multi-objetivo	81
Tabela 7 - Publicações que adotaram funções-objetivo estocásticas	86
Tabela 8 - Intervalos de manutenção (horas) recomendados pela empresa fabricante dos veículos.....	95
Tabela 9 - Dias de operação $\times$ taxas de deterioração (horas)	96
Tabela 10 - Configuração dos parâmetros do banco de problemas (por problema).....	112
Tabela 11 - Valores do parâmetro escala para problemas de grande porte	113
Tabela 12 - Valores do parâmetro escala para problemas de pequeno porte	114
Tabela 13 - Valores do parâmetro escala para base de dados para ajuste de parâmetros.....	114
Tabela 14 - Desempenho das heurísticas construtivas em relação ao custo total de MP para minimização de C_{max} no cenário YS... ..	143
Tabela 15 - Desempenho das heurísticas construtivas em relação ao custo total de MP para minimização de C_{max} no cenário RGM.....	144
Tabela 16 - Teste de honestidade de Tukey ao nível de 5% de significância para cenário YS	147
Tabela 17 - Teste de honestidade de Tukey ao nível de 5% de significância para cenário RGM	148
Tabela 18 - ANOVA de duas vias para parâmetros do IG no cenário YS	157
Tabela 19 – Teste de honestidade de Tukey ao nível de 5% de significância para IG no cenário YS	157
Tabela 20 - ANOVA de duas vias para parâmetros do IG no cenário RGM	158
Tabela 21 – Teste de honestidade de Tukey ao nível de 5% de significância para IG no cenário RGM.....	158
Tabela 22 – Teste de honestidade de Tukey ao nível de 5% de significância para a interação d * jP no cenário RGM.....	159

Tabela 23 – ANOVA ao nível de 5% de significância para DWWO no cenário YS.....	163
Tabela 24 – Teste de honestidade de Tukey ao nível de 5% de significância para DWWO no cenário RGM	167
Tabela 25 - ANOVA para interação entre pop * λ	168
Tabela 26 - Desempenho das metaheurísticas em comparação ao MILP no cenário YS	170
Tabela 27 - Desempenho das metaheurísticas em comparação ao MILP no cenário YS	171
Tabela 28 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário YS	173
Tabela 29 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário RGM	173
Tabela 30 - Desempenho das metaheurísticas para problemas de médio e grande porte agrupados por $n \times m$ no cenário YS	177
Tabela 31 - Desempenho das heurísticas para problemas de médio e grande porte agrupados por $n \times m$ no cenário RGM.....	178
Tabela 32 - Teste de honestidade de Tukey para problemas de médio e grande porte no cenário YS	179
Tabela 33 - Teste de honestidade de Tukey para problemas de médio e grande porte no cenário RGM	179
Tabela 34- Desempenho das heurísticas construtivas para minimização do custo total de manutenção preventiva e tempo total de fluxo no cenário YS	202
Tabela 35- Desempenho das heurísticas construtivas para minimização do custo total de manutenção preventiva e tempo total de fluxo no cenário RGM	203
Tabela 36 - Teste de honestidade de Tukey ao nível de 5% de significância para as heurísticas construtivas no cenário YS	205
Tabela 37 - Teste de honestidade de Tukey ao nível de 5% de significância para as heurísticas construtivas no cenário RGM	205
Tabela 38 - Tempo computacional (em segundos) das heurísticas construtivas no cenário YS	209
Tabela 39 - Tempo computacional (em segundos) das heurísticas construtivas no cenário RGM	209
Tabela 40 - ANOVA para análise dos parâmetros adotados em HPF2(GRASP) no cenário YS	212
Tabela 41 - ANOVA para análise dos parâmetros adotados em HPF2(GRASP) no cenário RGM	212

Tabela 42 - Faixas de valores para ($\min\mu$; $\max\mu$) nos cenários YS e RGM	214
Tabela 43 - Resultados da ANOVA ao nível de 5% de significância dos parâmetros necessários para DABC_RCT no cenário YS.....	215
Tabela 44 - Resultados da ANOVA dos parâmetros necessários para DABC_RCT no cenário YS	215
Tabela 45 - Resultados da ANOVA ao nível de 5% de significância dos parâmetros necessários para VBIH no cenário YS.....	217
Tabela 46 - Resultados da ANOVA ao nível de 5% de significância dos parâmetros necessários para VBIH no cenário RGM.....	217
Tabela 47 - Desempenho das metaheurísticas para problemas de pequeno porte no cenário YS, com desvios relativos médios agrupados por $n \times m$	221
Tabela 48 - Desempenho das metaheurísticas para problemas de pequeno porte no cenário RGM, com desvios relativos médios agrupados por $n \times m$	222
Tabela 49 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário YS	222
Tabela 50 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário RGM	223
Tabela 51 - Desvio relativo médio agrupados por $n \times m$ das metaheurísticas no cenário YS.....	227
Tabela 52 - Desvio relativo médio agrupados por $n \times m$ das metaheurísticas no cenário RGM	228
Tabela 53 - Teste de honestidade de Tukey ao nível de 5% de significância para metaheurísticas em problemas de médio e grande porte no cenário YS	230
Tabela 54 - Teste de honestidade de Tukey ao nível de 5% de significância para metaheurísticas em problemas de médio e grande porte no cenário RGM	230
Tabela 54 - Desvios relativos para todas as combinações entre μ e λ da heurística HPF2 no cenário YS	261
Tabela 55 - Desvios relativos para todas as combinações entre μ e λ da heurística HPF2 no cenário RGM	262

LISTA DE SIGLAS E ABREVIATURAS

ABC	<i>Artificial Bee Colony</i>
AIS	<i>Artificial Immune System</i>
B&B	<i>Branch & Bound</i>
CGLS	<i>Constraint Guided Local Search</i>
CRO	<i>Chemical Reaction Optimization</i>
CSA	<i>Clustering Search Algorithm</i>
CuckooSA	<i>Cuckoo Search Algorithm</i>
DE	<i>Differential Evolution</i>
DGSO	<i>Multi-objective discrete group search optimizer</i>
DRM	Desvio relativo médio
DWWO	<i>Discrete Water Wave Optimization</i>
EDA	<i>Estimation Distribution Algorithm</i>
FFOA	<i>Fruit Fly Optimization Algorithm</i>
FIFO	Ordenação por <i>First-In First-Out</i>
GA	<i>Genetic Algorithm</i>
GRASP	<i>Greedy Randomized Adaptive Search Procedure</i>
GWO	<i>Grey Wolf Optimization</i>
HS	<i>Harmony Search</i>
IBBO	<i>Improved Biography-Based Optimization Algorithm</i>
IG	<i>Iterated Greedy</i>
ILS	<i>Iterated Local Search</i>
IWO	<i>Invasive Weed Optimization</i>
MA	Atividade de Manutenção Preventiva
MBO	<i>Migrating Birds Optimization</i>
MMA	<i>Memetic Algorithm</i>
MC	Manutenção Corretiva
MDGSO	<i>Multi-objective discrete group search optimizer</i>
Meta-RaPS	<i>Metaheuristic for Random Priority Search</i>
MILP	Programação Linear Inteira Mista
MOEA	<i>Multi-Objective Evolutionary Algorithm</i>

MP	Manutenção Preventiva
NM	Nível de manutenção
PSO	<i>Particle Swarm Optimization</i>
REMO	<i>Robust Evolutionary Scheduling Algorithm</i>
RGM	Cenário de manutenção preventiva de Ruiz et al. (2007)
SA	<i>Simulated Annealing</i>
SOMA	<i>Self-organising Migrating Algorithm</i>
SSA	<i>Scatter Search</i>
TPA	<i>Three Phase Algorithm</i>
TS	<i>Tabu Search</i>
VBIH	<i>Variable Block Insertion Heuristic</i>
VNS	<i>Variable Neighborhood Search</i>
YS	Cenário de manutenção preventiva de Yu e Seif (2016)

SUMÁRIO

1 INTRODUÇÃO	29
1.1 MOTIVAÇÃO E JUSTIFICATIVA DA PESQUISA	33
1.2 OBJETIVO DA PESQUISA E ESTRUTURAÇÃO DA TESE	34
2 REVISÃO DE LITERATURA.....	37
2.1 NOTAÇÕES E TERMINOLOGIAS DE <i>SCHEDULING</i>	37
2.1.1 Métodos de Solução	42
2.2 ESTRUTURAÇÃO DA REVISÃO DE LITERATURA.....	44
2.3 FUNÇÕES-OBJETIVO BASEADAS EM DATA DE TÉRMINO.....	46
2.3.1 Duração total da programação (C_{max}).....	46
2.3.1.1 <i>Flow shop</i> com bloqueio e minimização de C_{max}	51
2.3.1.2 Problema <i>flow shop</i> com bloqueio, restrições adicionais e minimização de C_{max}	64
2.3.1.3 Problema <i>flow shop</i> com diferentes tipos de bloqueio e minimização de C_{max}	68
2.3.1.4 Problema <i>flow shop</i> com <i>buffers</i> limitados e minimização de C_{max}	69
2.3.2 Problema <i>flow shop</i> com bloqueio e minimização de tempo total de fluxo	72
2.3.3 Problema <i>flow shop</i> com bloqueio e função-objetivo baseadas em datas de entrega	76
2.3.4 Problema <i>flow shop</i> com bloqueio e minimização do tempo de ciclo.....	79
2.3.5 Problema <i>flow shop</i> com bloqueio e função multi-objetivo	80
2.3.6 Problema <i>flow shop</i> com bloqueio e minimização de função-objetivo estocástica ..	85
2.3 CONSIDERAÇÕES FINAIS DA SEÇÃO 2	88
3 DEFINIÇÃO DO PROBLEMA	91
3.1 PROBLEMA DE <i>SCHEDULING FLOW SHOP</i> COM BLOQUEIO E <i>SETUP</i> DEPENDENTE DA SEQUÊNCIA.....	91
3.2 MANUTENÇÃO PREVENTIVA BASEADA EM NÍVEIS.....	92
3.3 <i>SCHEDULING</i> E MANUTENÇÃO PREVENTIVA	94
3.3.1 Modelo matemático proposto para os problemas com minimização de duração total da programação e tempo total de fluxo	98
3.3.2 Cálculo das funções-objetivo	102
3.4 BUSCA LOCAL PARA PROGRAMAÇÃO E REPOSICIONAMENTO DAS MANUTENÇÕES PREVENTIVAS PARA MINIMIZAÇÃO DA FUNÇÃO-OBJETIVO	105
3.5 INTERVALOS ENTRE MANUTENÇÕES PREVENTIVAS.....	110

3.6 CONFIGURAÇÕES DAS INSTÂNCIAS DE PROBLEMAS PARA EXPERIMENTOS COMPUTACIONAIS	111
4 MÉTODOS DE SOLUÇÃO PARA O PROBLEMA DE MINIMIZAÇÃO DA DURAÇÃO TOTAL DA PROGRAMAÇÃO SUJEITA AOS CUSTOS DE MANUTENÇÃO PREVENTIVA	117
4.1 HEURÍSTICAS CONSTRUTIVAS PARA MINIMIZAÇÃO DO CUSTO TOTAL DE MANUTENÇÃO PREVENTIVA.....	117
4.1.1 Profile Fitting (PF)/PFE/PF-NEH(x)/PF-NEH(x)st.....	119
4.1.2 MinMax (MM)/MME, MM-B/MME-B	121
4.1.3 wPF	122
4.1.4 PW.....	123
4.1.5 PF-NEH(x)/PF-NEH(x)st, wPF-NEH(x)/wPF-NEH(x)st e PW-NEH(x)/PW-NEH(x)st.....	125
4.1.6 Modified NEH (mNEH)/mNEHst	127
4.1.7 WPT-NEH/WPT-NEH_{st}.....	128
4.1.8 PFT-NEH(x)/PFT-NEH(x)st e PFT-RZ(x)/PFT-RZ(x)st.....	128
4.1.9 PF+NEH(λ)/ PF+NEH(λ)st	130
4.2 METAHEURÍSTICAS	131
4.2.1 Iterated Greedy (IG)	131
4.2.2 Discrete Water Wave Optimization (DWWO)	134
4.2.2.1 Propagação	136
4.2.2.2 Refração.....	138
4.2.2.3 Quebra	139
4.2.2.4 Atualização da população	140
4.3 EXPERIMENTAÇÕES COMPUTACIONAIS	141
4.3.1 Análise de desempenho das heurísticas construtivas	141
4.3.2 Calibração de parâmetros das metaheurísticas IG E DWWO	156
4.3.2.1 Parametrização de <i>Iterated Greedy (IG)</i>	156
4.3.2.2 Parametrização de <i>Discrete Water Wave Optimization Algorithm (DWWO)</i>	161
4.3.3 Análise dos resultados das metaheurísticas para minimização de C_{max} para problemas de pequeno porte.....	169
4.3.4 Análise dos resultados para problemas de grande porte	176
4.4 CONSIDERAÇÕES FINAIS DA SEÇÃO 4	183

5 MÉTODOS DE SOLUÇÃO PARA O PROBLEMA DE MINIMIZAÇÃO DE TEMPO TOTAL DE FLUXO SUJEITO AO CUSTO TOTAL DE MANUTENÇÃO PREVENTIVA.....	185
5.1 HEURÍSTICAS CONSTRUTIVAS PARA MINIMIZAÇÃO DO TEMPO TOTAL DE FLUXO DADO O CUSTO TOTAL DE MANUTENÇÃO PREVENTIVA	185
5.1.1 HPF1, HPF2, NHPF1 e NHPF2.....	186
5.1.2 tPF-NEH(x)	189
5.2 METAHEURÍSTICAS PARA MINIMIZAÇÃO DO TEMPO TOTAL DE FLUXO	191
5.2.1 <i>Greedy Randomized Adaptive Search Procedure (GRASP)</i>	192
5.2.2 <i>Discrete Artificial Bee Colony (DABC)</i>	194
5.2.2.1 Inicialização.....	194
5.2.2.2 Abelhas Campeiras	195
5.2.2.3 Abelhas Observadoras	196
5.2.2.4 Abelhas exploradoras	196
5.2.3 <i>Variable Block Insertion Heuristic (VBIH)</i>	197
5.3 RESULTADOS COMPUTACIONAIS	200
5.3.1 Avaliação computacional das heurísticas construtivas	200
5.3.2 Calibração de parâmetros das metaheurísticas GRASP(HPF2), ABC_RCT e VBIH	211
5.3.2.1 Parametrização de GRASP(HPF2)	211
5.3.2.1 Parametrização de DABC_RCT	214
5.3.2.1 Parametrização de VBIH	217
5.3.3 Análise dos resultados das metaheurísticas para minimização do tempo total de fluxo para problemas de pequeno porte.....	220
5.3.4 Análise dos resultados das metaheurísticas para minimização do tempo total de fluxo para problemas de médio e grande portes	227
5.4 CONSIDERAÇÕES FINAIS DA SEÇÃO 5	233
6 CONCLUSÕES.....	235
6.1 SUGESTÃO PARA PESQUISAS FUTURAS	238
REFERÊNCIAS	239
APÊNDICE A – DESVIOS RELATIVOS DA HEURÍSTICA HPF2 PARA OS CENÁRIOS YU E SEIF	261

1 INTRODUÇÃO

O problema de programação de operações ou *scheduling*, é um processo de decisão que desempenha um papel crucial em muitas indústrias de manufatura e de serviços. Em ambientes empresariais competitivos, as decisões de *scheduling* impactam diretamente no atendimento dos clientes em prazos corretos e no uso mais eficiente dos recursos disponíveis (PINEDO, 2008). A programação de operações é uma das atividades mais complexas no gerenciamento da produção, pois é de atribuição dos programadores determinar quais recursos serão alocados para processar bens ou serviços, assim como determina quando esses bens e serviços serão processados, ou seja, sua essência concentra-se em decisões de alocação e sequenciamento (SLACK et al., 1996).

Um problema de *scheduling* consiste em encontrar uma sequência de n tarefas (bens e serviços) que serão processadas em um conjunto de m máquinas (equipamentos, colaboradores, máquinas propriamente ditas) com o objetivo de minimizar alguma medida adotada (BAKER, 1974). O problema de sequenciamento varia de acordo com o ambiente de produção onde ocorre o processamento das tarefas, o critério de desempenho adotado e as restrições adicionais que podem advir das tarefas ou dos recursos disponíveis (GRAHAM et al., 1979).

Um dos ambientes de produção clássicos mais estudados desde a década de 50 é o denominado *Flow shop Permutacional* (MACCARTHY; LIU, 1993; REZA HEJAZI; SAGHAFIAN, 2005; GUPTA; STAFFORD JR, 2006). O problema de *scheduling* em um *flow shop* permutacional consiste em sequenciar n tarefas que devem ser processadas em m máquinas em série. Todas as tarefas possuem o mesmo fluxo padrão de produção, ou seja, passam pela mesma ordem processamento nas máquinas. A restrição permutacional implica que a sequência de tarefas estabelecida na primeira máquina é a mesma para todas as máquinas restantes.

As restrições adicionais dizem respeito às características das tarefas ou dos recursos disponíveis e permitem uma aproximação maior entre a teoria de *scheduling* e a realidade prática (MACCARTHY; LIU, 1993; HALL; SRISKANDARAJAH, 1996). Nesta pesquisa considera-se três restrições adicionais ao problema *flow shop* permutacional: bloqueio; tempos de *setup* e; uma abordagem de manutenção preventiva (MP). Na Figura 1 apresenta-se um exemplo ilustrativo em gráfico de *gantt* da ocorrência das três restrições adicionais incorporadas ao *flow shop* permutacional.

Figura 1 - *Flow shop* permutacional com 4 tarefas e 4 máquinas

Fonte: Autor, 2019.

O tempo/custo de *setup* é uma das restrições clássicas do problema de *scheduling* (ALLAHVERDI et al., 1999, ALLAHVERDI et al., 2008, ALLAHVERDI, 2015). O *setup* ou preparação de máquina compreende as medidas necessárias para se preparar a máquina para processar determinada tarefa. Da perspectiva apresentada por Allahverdi et al. (2008), o *setup* pode ser especificado de acordo com:

- A dependência da sequência das tarefas – o tempo de *setup* pode variar de acordo com sua tarefa adjacente predecessora, o que se denomina como *setup* dependente da sequência. Na Figura 1, $S_{i,j}$ representa o *setup* da tarefa j que depende da tarefa i . Em outro caso, os tempos de *setup* das tarefas são diferentes entre si, porém, independente da posição que esta tarefa ocupe, a tarefa adjacente predecessora não afeta sua duração, ou seja, o *setup* independente da sequência;
- A antecipação de sua operação – Se as especificações do produto ou se o ambiente de produção permitir ou se não há custos significativos envolvidos, o tempo de *setup* para a próxima tarefa pode ser efetuado a qualquer momento a partir de quando a atual tarefa sai da máquina (Veja exemplo na Figura 1). Neste caso, denomina-se *setup* antecipatório e caso contrário, se deve ser realizado ao processar a tarefa, o *setup* é denominado não-antecipatório.

Outra restrição que vem ganhando notoriedade nas últimas décadas é a condição de bloqueio (MIYATA; NAGANO, 2019). Este problema ocorre devido à ausência de estoques intermediários entre as máquinas, às características das tarefas ou às restrições tecnológicas do

processo (HALL; SRISKANDARAJAH, 1996). Na Figura 1, as tarefas 2 e 3 permanecem nas respectivas máquinas 1 e 3, até que seus tempos de *setup* sejam concluídos nas próximas máquinas. Devido a isso, as tarefas 2 e 3 impedem as máquinas 1 e 3 de realizar o processamento de outras atividades gerando um tempo de ociosidade, definido como o termo “bloqueio”.

A vasta maioria dos problemas de *scheduling* consideram a hipótese de que a máquina está sempre disponível para processar as tarefas. Entretanto, sabe-se que em ambientes industriais, as máquinas estão sujeitas a quebras. Recentemente, pesquisas têm sido direcionadas a estudar os aspectos envolvidos com a hipótese de disponibilidade da máquina (MA et al., 2010). A máquina pode se tornar indisponível devido a uma quebra de máquina ou a uma manutenção preventiva (MP) durante o período de programação. Quando a indisponibilidade é dada por MP, dois casos são possíveis. Em um dos casos, o período de manutenção é conhecido e fixo, o que se denomina como caso determinístico. Em outro caso, os períodos de manutenção também são variáveis de decisão, ou seja, a programação das tarefas e a manutenção da máquina são feitas simultaneamente. Quando as indisponibilidades ocorrem devido à quebra de máquinas, denomina-se caso estocástico, as quebras e reparos são aleatórias (MA et al., 2010). Nesta pesquisa, é considerado o primeiro caso. Como decisão de sequenciamento, a MP é vista como uma atividade com uma duração significativa que deve ser realizada em um intervalo específico de tempo. Na literatura da integração entre MP e *scheduling*, o intervalo entre MP's é abordado pela teoria estatística da confiabilidade (CASSADY; KUTANOGLU, 2003, SORTRAKUL et al., 2005, RUIZ et al., 2007, ZANDIEH; GHOLAMI, 2009, WANG; LIU, 2013, TSANG et al., 2014, WANG; LIU, 2014b, KHAMSEH et al., 2015, MIYATA et al., 2019), por um dado período conhecido (ALLAOUI; ARTIBA, 2004, ASHAYERI et al., 1996, GRAVES; LEE, 1999, QI et al., 1999, LEE; CHEN, 2000, AGGOUNE, 2004, RAZA et al., 2007, HADDA et al., 2008, CHOI et al., 2010, SUN; LI, 2010, KHELIFATI; BENBOUZID-SITAYEB, 2011, SITAYEB et al., 2011, CHIU; SHIH, 2012, REBAI, M. et al., 2013, VAHEDI-NOURI et al., 2014, LUO; JI, 2015, LUO et al., 2015, XU et al., 2015), por frequências de atividades de MP durante o horizonte de programação (YANG, 2012, HNAIEN; YALAOUI, 2013, VAHEDI-NOURI et al., 2013, ZAROOK et al., 2014, CHEN, 2008, MOSHEIOV; SARIG, 2009b), por meio de manutenção baseada em condição (SAFARI et al., 2010, XU et al., 2010, BOCK et al., 2012, YANG, 2013, YU; SEIF, 2016, MIYATA et al., 2019), por funções lineares que estabelecem a duração da manutenção (CHENG et al., 2012, YANG; YANG, 2010) ou quando a velocidade de processamento da máquina reduz proporcionalmente em relação ao estado de uso da máquina (LEE; LIN, 2001,

MOSHEIOV; SARIG, 2009a, ZHAO; TANG, 2010, MOR; MOSHEIOV, 2012). Após a realização da atividade de MP, a máquina retorna ao seu estado de operação em condições parciais operáveis ou então na condição “tão boa quanto nova”. Neste trabalho, a abordagem de MP é baseada na condição da máquina. Cada máquina possui níveis de MP em que cada tarefa e seu respectivo *setup* deterioram a máquina por meio de uma taxa linear. A medida que as tarefas são processadas, um determinado nível de MP t da máquina vai sendo decrescido até atingir um certo limite estabelecido. Então, uma atividade de MP é realizada para o nível t para que a máquina volte a um estado de condições operáveis.

As medidas de desempenho consistem na função-objetivo que se pretende minimizar. Estas medidas estão relacionadas com a melhor utilização eficiente dos recursos disponíveis, como por exemplo, a minimização da duração total da programação, a redução de custos de inventário, como a minimização do tempo total de fluxo das tarefas no processo e, o atendimento ao cliente, como a minimização do número de tarefas atrasadas ou do atraso total (MACCARTHY; LIU, 1993). O problema de scheduling *flow shop* que focam em um único objetivo tem sido objeto de estudo de muitos pesquisadores ao longo dos anos. Uma gama de métodos de solução foram propostos para tais. Uma vez que a maioria dos problemas de *scheduling* na prática envolvem múltiplos objetivos, o *flow shop* com funções multi-objetivo tem ganhado notoriedade nos últimos anos (YENISEY; YAGMAHAN, 2014). De acordo com Emmons e Vairaktarakis (2013a), há três formas de formular uma função-objetivo multicritério: a) funções hierárquicas, onde uma função A é minimizada dada que a função B deve ser menor ou igual a um determinado valor C ; b) funções compostas, onde a combinação linear de A e B são minimizadas e; c) função multicritério, que fornece a curva de pareto dos melhores valores encontrados para A e B , apresentados como as soluções do problema para o tomador de decisão. Nesta pesquisa é adotada a função hierárquica, que expressa a minimização da duração total da programação (C_{max}) ou do tempo total de fluxo ($\sum_i C_i$), ambos sujeitos ao custo total de MP. Uma revisão de literatura recente em *flow shop* com funções multicritério pode ser vista em Yenisey e Yagmahan (2014). A abordagem de funções hierárquicas para o problema *flow shop* inicia com a publicação de Daniels e Chambers (1990), que propuseram métodos de solução para o problema de minimização da combinação convexa entre duração total da programação e o atraso máximo. Posteriormente, Chakravarthy e Rajendran (1999) e Framinan e Leisten (2006), abordam a minimização da duração total da programação dado um atraso máximo. Recentemente, as pesquisas de Aydilek e Allahverdi (2012), Allahverdi e Aydilek (2014), Allahverdi et al. (2018) e Allahverdi et al. (2020) adotaram as funções hierárquicas aplicados ao problema *no-wait flow shop*.

Nos últimos anos, os métodos de solução para o problema de *scheduling* têm tomado uma abordagem cada vez mais robusta e eficiente. Para um pequeno número de tarefas e máquinas, métodos exatos têm sido utilizados com sucesso. Entretanto, dependendo do ambiente de produção, à medida que são incorporadas restrições adicionais e quando o número de tarefas e máquinas aumenta, torna-se inviável a utilização de métodos exatos para o problema. Para $m > 2$ máquinas e minimização da duração total da programação, o problema *flow shop* com bloqueio foi provado ser da classe de problemas NP-Difícil (HALL; SRISKANDARAJAH, 1996). Neste sentido, o emprego de heurísticas, métodos que geram soluções próximas da ótima, tem sido foco das pesquisas recentes.

1.1 MOTIVAÇÃO E JUSTIFICATIVA DA PESQUISA

Uma das grandes dificuldades pontuadas por Maccarthy e Liu (1993) em relação ao *scheduling* é a grande distância que há entre realidade e teoria. A adição de novas hipóteses advindas da realidade prática pode diminuir este *gap* com os modelos teóricos. Neste contexto, como já pontuado anteriormente, esta pesquisa adota três situações frequentemente presentes em ambientes reais: tempos de *setup* dependentes da sequência, manutenção preventiva e bloqueio entre estágios produtivos.

O *flow shop* com bloqueio pode ser visto em diversos ambientes produtivos tais como na indústria química, produção de peças aeronáuticas (MARTINEZ et al., 2006), na produção de cidra (TRABELSI et al., 2012), de placas de circuitos eletrônicos (GONG et al., 2010), de ferro e aço (HALL; SRISKANDARAJAH, 1996), de blocos de concreto (GRABOWSKI; PEMPERA, 2000), no tratamento de dejetos (MARTINEZ et al., 2006) e também no setor de serviços como em operações de terraplanagem (YU; SEIF, 2016).

A incorporação do tempo de *setup* em problemas de *scheduling* tem sido estudada desde a década de 60. Allahverdi et al. (1999), Allahverdi et al. (2008), Allahverdi e Soroush (2008) e Allahverdi (2015) destacam a relevância desta restrição ao problema de *scheduling* e também em situações reais. Na prática, as considerações de *setup* dependente da sequência impactam diretamente na melhor utilização de recursos. Os benefícios da redução do tempo de *setup* incluem a redução de custos, aumento da velocidade da produção, redução de *lead times* e maior agilidade no processo de troca de ferramentas e na entrega do produto ao cliente, aumentando sua satisfação.

Segundo Ruiz *et al.* (2007) e Ma *et al.* (2010) a manutenção preventiva (MP) impacta diretamente no tempo disponível para produção, assim como também, torna mais real a hipótese da disponibilidade da máquina. Na prática, as decisões de MP e de programação de operações são realizadas independentemente apesar do relacionamento direto entre elas. Atividades de MP consomem tempos de produção potenciais. Por outro lado, adiar uma MP por causa da demanda da produção pode aumentar o risco de probabilidade de falha de máquina. Portanto, integrar estas duas decisões pode vir a trazer benefícios na tomada de decisões.

Apesar do grande destaque recebido nas últimas décadas, como poderá ser visto na Seção 2, a maioria das pesquisas voltam-se para a concepção de métodos do problema *flow shop* com bloqueio sem restrições adicionais e com adoção de funções mono-objetivo. Estes resultados indicam um grande campo de pesquisas e estudos na área. Neste sentido, pela revisão de literatura realizada, até então, não foram encontradas pesquisas que tratam do problema apresentado.

1.2 OBJETIVO DA PESQUISA E ESTRUTURAÇÃO DA TESE

O objetivo desta pesquisa é a concepção de métodos de solução para o problema *flow shop* com m máquinas, ocorrência de bloqueio, de tempos de *setup* dependente da sequência das tarefas e de atividades de manutenção preventiva (MP) para minimização da duração total da programação e do tempo total de fluxo, dado que o custo total da MP não pode ser maior que um determinado valor. Os seguintes objetivos específicos compõem o objetivo geral:

- Revisão e classificação das publicações existentes de *scheduling* para o problema *flow shop* com restrição de estoque intermediário;
- Estudo e modelagem matemática do problema anteriormente referido, bem como formulações e formas de cálculo das funções-objetivo adotados;
- Desenvolvimento de procedimentos para programação e reposicionamento das atividades de MP's na sequência de tarefas;
- Proposição de técnicas heurísticas para o problema abordado com a minimização da duração total da programação do problema e do tempo total de fluxo, dado que o custo de MP não pode ser maior que um determinado valor.

Este trabalho estrutura-se da seguinte forma: Na Seção 2 apresenta-se o referencial teórico básico sobre as notações e terminologias básicas de *scheduling*, assim como a definição dos métodos de solução encontrados durante a revisão de literatura. Em seguida, apresenta-se a revisão de literatura e a classificação dos métodos de solução para o problema *flow shop* com bloqueio. A terceira Seção dedica-se à definição dos problemas de *scheduling* apresentados anteriormente, mostrando as hipóteses consideradas, os intervalos entre MP's e as formas de cálculo da duração total da programação e do tempo total de fluxo com a inclusão das atividades de MP. Uma busca local para reposicionamento das MP's também é proposta. Nas Seções 4 e 5 descrevem-se os métodos de solução adaptados e propostos para os problemas *flow shop* com bloqueio, *setup* dependente da sequência, ocorrência de manutenção preventiva e minimização da duração total da programação e do tempo total de fluxo sujeito ao custo total de MP, respectivamente. São também apresentados resultados experimentais do desempenho dos métodos implementados para instâncias de problemas de pequeno, médio e grande porte. A sexta Seção apresenta as conclusões da tese, respondendo cada objetivo elencado nesta Seção. Por fim, podem ser vistas as referências bibliográficas e os apêndices.

2 REVISÃO DE LITERATURA

Nesta Seção são apresentados os resultados da revisão de literatura para o problema *flow shop* com bloqueio. Foram cobertas publicações que dataram de 1969 à Maio de 2019, no intuito de identificar o estado da arte nesta área, além de levantar os métodos de solução propostos para os diversos problemas encontrados duante a revisão. O problema de *scheduling* com tempos de *setup* foi amplamente revisado e *surveys* podem ser vistos em Yang e Liao (1999), Allahverdi et al. (1999), Allahverdi et al. (2008) e Allahverdi (2015). Por se tratarem de *surveys* atualizados, para maior conhecimento sobre a literatura com considerações de *setup*, o leitor pode pesquisar as referidas publicações.

Esta seção se subdivide em duas seções principais. Na primeira seção, são apresentadas as notações utilizadas no decorrer da revisão. Na segunda seção, é apresentada a revisão de literatura do problema *flow shop* com diferentes ambientes com bloqueio, com e sem restrições adicionais e com diferentes funções objetivo.

2.1 NOTAÇÕES E TERMINOLOGIAS DE *SCHEDULING*

O problema *flow shop* com bloqueio pode ser definido como segue: Há n tarefas que devem ser processadas por meio de m máquinas. O processamento da tarefa i na máquina j é denotado como O_{ij} . Cada operação i ($i = 1, \dots, n$) possui um tempo de processamento associado com cada máquina j ($j = 1, \dots, m$), é denotado como $p_{i,j}$. Os estoques intermediários (*buffers*) entre as máquinas não existem, ou seja, a capacidade do estoque é considerada zero. Consequentemente, uma tarefa finalizada na máquina j permanece ali até que a máquina subsequente esteja disponível para processá-la. A literatura específica identifica dois tipos diferentes de ocorrência de bloqueio:

- a) Liberação da tarefa quando o bloqueio inicia (*block*): A máquina anterior permanece bloqueada pela tarefa i até que ela seja iniciada na próxima máquina do fluxo de produção. Em aplicações práticas, a ocorrência da condição *block* pode ser vistas em processos de produção de blocos de concreto (GRABOWSKI; PEMPERA, 2000), células robóticas (Carlier et al., 2010), indústria de ferro e aço (Gong, Tang, & Duin, 2010), manufatura de eletrônicos (Chen et al., 2014), entre outros;

- b) Liberação da tarefa quando o bloqueio termina (RCb^*/RCb) a máquina estará disponível para processar a próxima tarefa após a tarefa anterior deixar a máquina subsequente. A restrição RCb^* pode ser vistas em linhas de produção em que duas máquinas subsequentes dependem dos mesmos recursos, o que impede que elas sejam processadas ao mesmo tempo. Em um caso particular de RCb^* , denominado RCb , a máquina estará disponível para a próxima tarefa após a tarefa anterior ter sido finalizada e liberada na máquina subsequente. Exemplos das condições RCb e RCb^* podem ser vistas na fabricação de peças aeronáuticas e indústrias de tratamento de dejetos (MARTINEZ et al., 2006) e em operações de produção de cidra, apresentado por Trabelsi et al. (2012);

No problema *flow shop* com bloqueio entre as m máquinas, a data de término de uma tarefa i na máquina j é dada pela sua data de partida $D_{i,j}$, calculada como segue:

$$D_{1,0} = 0, \quad (1)$$

$$D_{1,j} = \sum_{k=1}^j p_{1,k}, \quad k = 1, \dots, m-1 \quad (2)$$

$$D_{i,0} = D_{i-1,1}, \quad i = 2, \dots, n \quad (3)$$

$$D_{i,j} = \max(D_{i,j-1} + p_{i,j}, D_{i-1,j+1}), \quad i = 2, \dots, n \quad k = 1, \dots, m-1 \quad (4)$$

$$D_{i,m} = D_{i,m-1} + p_{i,m}, \quad i = 1, \dots, n \quad (5)$$

Deste modo, a duração total da programação ou *makespan* (C_{max}) corresponde à data de partida da última tarefa na última máquina:

$$C_{max} = D_{n,m} \quad (6)$$

Para o problema $F_m|block|C_{max}$, Wang et al. (2010) apresentam uma técnica de aceleração do cálculo de C_{max} para o caso de uma busca local de inserção com $(n-1)^2$ sequências geradas no problema. O procedimento reduz o tempo computacional de $O(mn^3)$ para $O(mn^2)$.

O tempo total de fluxo corresponde à soma das datas de partida de todas as tarefas na última máquina:

$$\sum_i C_i = \sum_{i=1}^n D_{i,m} \quad (7)$$

Fernandez-Viagas *et al.* (2016) apresentam procedimentos de aceleração do cálculo de $\sum_i C_i$ no problema $F_m|block|\sum_i C_i$ para a busca local de inserção. Tasgetiren *et al.* (2016) também propõem um procedimento de aceleração para o cálculo de $\sum_i C_i$ para buscas locais de inserção e troca entre pares de tarefas. Ambos os autores se baseiam no procedimento de aceleração de Li *et al.* (2009), que mostrou reduzir de 40 a 50% o tempo computacional.

Segundo Maccarthy e Liu (1993), um problema de programação das operações pode ser classificado de acordo com o ambiente de máquinas, as restrições em relação às máquinas e às tarefas e o critério de desempenho considerado. Graham *et al.* (1979) define um problema de programação de operações por meio de uma notação de três campos $\alpha|\beta|\gamma$.

O campo $\alpha = \alpha_1\alpha_2$ especifica as restrições tecnológicas ou ambientes de máquinas, sendo α_1 o ambiente de máquina considerado e α_2 a quantidade de máquinas ou estágios do ambiente. Os ambientes de produção clássicos podem ser vistos em Graham *et al.* (1979). No ambiente *flow shop* com m máquinas, denotado por $\alpha_1\alpha_2 = F_m$, cada tarefa é processada em cada uma das m máquinas que estão dispostas em série. Todas as tarefas têm que seguir o mesmo fluxo padrão nas máquinas (PINEDO, 2008).

O campo $\beta \in \{\beta_1, \dots, \beta_n\}$ representa as condições ou restrições adicionais relacionadas às características das tarefas ou do ambiente de máquinas (GRAHAM *et al.*, 1979). Estas restrições incluem:

- Tempos de *setup* – Refere-se ao tempo incorrido para preparar uma máquina para processar uma tarefa incluindo a obtenção de ferramentas, o trabalho de posicionar o material em processo, limpeza, ajuste de ferramentas, inspeção de materiais e definir peças e acessórios necessários (ALLAHVERDI *et al.*, 2008). O tempo de *setup* dependente da sequência que ocorre entre as tarefas i e k pode ser denotado como $s_{i,k}$. Se o tempo de *setup* entre as tarefas i e k dependem da máquina j , então o subscrito j é incluído, ou seja, $s_{j,i,k}$. Caso o *setup* não possa ser antecipado, o subscrito *non-ant* é adicionado ao campo β ;
- *No-wait* – As operações das tarefas não podem esperar por processamento entre duas máquinas sucessivas. Isso implica que a operação deve esperar na primeira máquina para assegurar que possa ser processada continuamente em todas as máquinas. Por exemplo, nas operações de laminadoras de aço, uma lâmina de aço não pode esperar para ser processada, pois iria esfriar durante a espera, causando perda de material ou custos de retrabalho (HALL; SRISKANDARAJAH, 1996);

- Datas de liberação (r_i) – Corresponde ao instante ou data em que a tarefa i está permitida para iniciar seu processamento (PINEDO, 2008);
- Preempção ($prmp$) – A preempção implica que não é necessário manter uma tarefa em uma máquina, uma vez iniciada, até o seu término. O programador pode interromper o processamento da tarefa a qualquer momento e iniciar o processamento de uma outra tarefa. Quando a tarefa interrompida retorna, a máquina processa somente o tempo restante. A inclusão do termo $prmp$ no campo β significa que o processo permite preempção, caso contrário, esta informação não é disponibilizada (PINEDO, 2008);
- Permutação – Ocorre quando a ordem das tarefas em todas as máquinas deve ser mantida, ou seja, o fluxo de processamento das tarefas nas máquinas será sempre o mesmo em todas as máquinas (PINEDO, 2008). Caso o ambiente de produção seja não permutacional, a notação *non-perm* é adicionada no campo β ;
- Estoques intermediários limitados ($b_{j,j+1}$): Ocorre quando o estoque intermediário entre as máquinas é finito, podendo variar de $0 \leq b_{j,j+1} \leq b$ (NORMAN, 1999);
- Datas de lançamento (r_i) – A tarefa i somente poderá ser processada após sua data de lançamento r_i (PINEDO, 2008);
- Lotes de tarefas (*batch*) – Tarefas similares ou famílias de tarefas são processadas em lotes (AGNETIS et al., 1997);
- Máquina múlti-tarefas ($mtflx(j)$): a máquina j pode realizar algumas operações de outras máquinas (KHORASANIAN; MOSLEHI, 2017);
- Número de caminhões (v): denota o número de caminhões idênticos utilizados para transportar as tarefas já processadas até os clientes (SOUKHAL et al., 2005);
- Capacidade do caminhão (c_i): Cada caminhão possui uma capacidade c_i associada ao número de tarefas que se pode transportar (SOUKHAL et al., 2005);
- *Flow shop* com entrega de tarefas ($F_m \rightarrow D$): as tarefas são primeiramente processadas em um *flow shop* (F_m) e, posteriormente, são carregadas em caminhões de transporte para serem entregues ao cliente (D) (SOUKHAL et al., 2005);
- Tarefas com taxas de deterioração (ap_{ij}): indica tarefas que estão sujeitas a deterioração ao longo do tempo. Um fator multiplicador associado ao seu tempo de processamento representa a taxa de deterioração (LEE et al., 2010);
- *Lot-streaming*(*lot-stream*): consiste em dividir a tarefa em um lote ou em vários sub-lotes de modo que sublotes já processados possam ser transportados para a máquina seguinte tal que sucessivas operações possam ser sobrepostas (HAN et al., 2016);

- *Flow shop* ordenado (*ord*): neste ambiente, as seguintes condições devem ser satisfeitas: se $p_{ij} < p_{kj}$ para quaisquer tarefas i e k na máquina j , então $p_{ij} \leq p_{kj}$ em todas as máquinas (tarefas ordenadas). Se $p_{ij} < p_{ig}$ para quaisquer duas máquinas j e g , então $p_{ij} \leq p_{ig}$ para todas as tarefas (máquinas ordenadas) (KOULAMAS; PANWALKAR, 2018);
- Quebra de máquinas (*brkdwn*) – Uma máquina pode falhar a qualquer momento de seu tempo operacional. Todas as máquinas estão sujeitas a falhas e um tempo de reparo é necessário para trazer a máquina para seu estado operacional novamente (HAN et al., 2019);
- Tempos de transferência (t_k): Tempo transcorrido para transportar a tarefa k da máquina anterior até a máquina atual (CARLIER et al., 2010b).

O campo γ especifica as funções-objetivo ou medidas de desempenho do problema (GRAHAM et al., 1979). As seguintes medidas de desempenho foram encontradas durante a revisão:

- Data de término da tarefa i ou *completion time* (C_i) - Data de término de processamento da tarefa i ($i = 1, \dots, n$). Se a data de término da tarefa i se refere a alguma máquina j ($j = 1, \dots, m$), então, $C_{i,j}$ representa a data de término da tarefa i na máquina j ;
- Duração total da programação determinístico/esperado ($C_{max} / E(C_{max})$): Data de término da última tarefa na última máquina;
- Tempo de Total de Fluxo determinístico/esperado ($\sum_i C_i / E(\sum_i C_i)$) ou tempo total de fluxo médio ($\sum_i C_i / n$) ou tempo total de fluxo médio ponderado ($\sum_i w_i C_i / n$): Soma dos instantes de término de todas as n tarefas. No caso do tempo de fluxo médio, o tempo total de fluxo é dividido pelas n tarefas.
- Atraso total das tarefas ($\sum_i T_i$) e atraso máximo determinístico/esperado ($T_{max} / E(T_{max})$): O atraso é definido como $T_i = \max(0, D_{i,m} - d_i)$. A soma de todos os atrasos é definida como atraso total das tarefas. O atraso máximo é definido como o maior atraso encontrado na sequência;
- Tempo de ciclo (CC): De acordo com McCormick et al. (1989), CC é dado por $\lim_{k \rightarrow \infty} \frac{b_{1,1}^k - b_{1,1}^1}{k-1}$, onde $b_{i,j}^k$ é a data de início da tarefa em uma estação j no conjunto de tarefas ou *mix* de produtos k ;

- Responsividade do cliente (CR): significa fornecer ao cliente o produto certo na quantidade certa, no prazo certo em um custo justo (COHN et al., 2010);
- Índice de aceitação (AI): Indica o grau de concordância entre a data de término da tarefa (C_i) e sua data de entrega (d_i) (YANG; LIU, 2018);
- Tempos total de espera entre as tarefas $\left(E\left(\sum_{i=1}^n \sum_{k=1}^n |w_{[i]}^2 - w_{[k]}^2|\right)\right)$ – Indica as diferenças em tempo de espera entre a data de término da tarefa i e a data de início da tarefa k na máquina seguinte (JIA, 1998);
- Consumo de energia (E_W): De acordo com Wang et al. (2018), o consumo total de energia em um *flow shop* é composto pelo consumo de energia para processar *setup*, transporte, máquinas ociosas, tarefas e uso público;
- Soma da ociosidade total ou bloqueio ($\sum_i (I_i)$ ou $\sum_i (B_i)$) – A ociosidade compreende o tempo que a máquina j espera pela finalização da tarefa i máquina $j - 1$.

Alguns exemplos de problemas, com o modelo de notação de Graham *et al.* (1979) podem ser vistos a seguir:

- $1|preemp, s_{i,j,k}|C_{max}$ denota um problema de máquina única com tempos de *setup* dependentes da sequência, se admite preempção das tarefas e o objetivo é minimizar a duração total da programação
- $F_m|block, s_{j,i,k}|\sum_i C_i$ – Indica o problema *flow shop* sem estoque intermediário entre as m máquinas, com tempos de *setup* dependentes da sequência e minimização do tempo total de fluxo;

2.1.1 Métodos de Solução

De modo geral, métodos de sequenciamento podem ser divididos em métodos exatos e métodos heurísticos. A Figura 2 evidencia a taxonomia dos métodos de solução.

Figura 2 - Taxonomia dos métodos de solução

Fonte: Autor, 2019.

Os métodos exatos encontrados durante a revisão para o problema foram a programação dinâmica, a técnica *branch & bound* e os modelos de programação linear inteira (MIP) e programação linear inteira mista (MILP). À medida que a complexidade do problema aumenta, métodos heurísticos têm sido empregados para gerar boas soluções e em tempo computacional viável. Os métodos heurísticos, de acordo com Framinan *et al.* (2004) podem ser classificados em construtivos e melhorativos.

Segundo Pinedo (2008), uma heurística construtiva fornece apenas uma sequência que é mantida como a solução do problema. Neste sentido, a sequência adotada pode ser gerada por meio de alguma ordenação baseada em índices de prioridade, escolhendo-se a melhor sequência a partir de um conjunto de sequências geradas ou ainda a partir da geração sucessiva de sequências parciais adotando-se uma estratégia de inserção de tarefas.

Métodos melhorativos partem de uma solução inicial e, por meio de algum procedimento iterativo, buscam melhores soluções que a atual para minimizar algum critério de desempenho. Neste caso, a solução inicial pode ser dada, por exemplo, por uma heurística construtiva ou por alguma simples regra de sequenciamento (FRAMINAN *et al.*, 2004). Para qualquer solução inicial gerada, busca-se sempre uma solução final melhor ou igual. Os métodos melhorativos mais empregados para solução de problemas de *scheduling* se concentram nas buscas locais e nas metaheurísticas.

Buscas locais partem de uma solução inicial s , que podem ser obtidas, por exemplo, por um método heurístico construtivo. A solução s associa-se a uma vizinhança N de s que pode ser encontrada por algum mecanismo de busca como inserção, troca entre pares de tarefas, entre outros (FINK; VOß, 2003). Cada solução s' , chamada de solução vizinha da vizinhança $N(s)$ é visitada a partir de s por meio de um movimento. Seleciona-se uma solução vizinha que seja

melhor que a solução corrente, e a busca prossegue iterativamente até que a vizinhança não contenha nenhuma solução melhor que a corrente ou se algum critério de parada tenha sido satisfeito (GRABOWSKI; PEMPERA, 2007).

Uma metaheurística pode ser formalmente definida como um processo iterativo que guia uma heurística subordinada combinando diferentes conceitos para explorar o espaço de busca. Estratégias de aprendizado são usadas para estruturar as informações para encontrar eficientemente soluções próximas da ótima. Estas técnicas são baseadas em conceitos advindos da evolução biológica, da matemática e da física, do sistema nervoso e da estatística mecânica (OSMAN; LAPORTE, 1996). Ao longo deste capítulo, as metaheurísticas encontradas durante a revisão da literatura serão brevemente descritas nas seções pertinentes.

2.2 ESTRUTURAÇÃO DA REVISÃO DE LITERATURA

Nesta seção, os termos e notações descritos na Seção 2.1 serão usados para especificar as publicações encontradas. O processo de busca levou em conta artigos que consideravam os problemas *flow shop* com m máquinas e com estoques intermediários ou *buffers* iguais a zero. Do total encontrado, apenas artigos escritos em língua inglesa foram considerados. Artigos que estudaram o problema *flow shop* com *buffers* limitados, onde a análise do problema cobriu o caso $b = 0$, também foram inclusos. Publicações que consideraram *flow shop* flexível, *flow shop* híbrido e *flow shop* distribuído não foram incluídos. A revisão foi realizada em base temporal, ou seja, a partir da primeira publicação encontrada até a mais recente (Maio de 2019). As bases de dados *ISI web of knowledge* e *Scopus* foram escolhidas para busca e filtragem de publicações. A Tabela 1 mostra as palavras-chave, resultados totais encontrados para cada conjunto de palavras-chave em cada banco de dados e os resultados filtrados, respectivamente.

Tabela 1 - Resultado das buscas nas bases de dados para problemas *flow shop* com bloqueio

Base de dados	Palavras-chave	Total Encontrado	Resultados filtrados
ISI Web of Knowledge	“blocking flow shop” or “flowshop with blocking” or “blocking flowshop” or “flow shop with blocking”.	120	90
	“flow shop with limited buffers” or “flowshop with limited buffers” or “flow shop with no intermediate storage” or “flow shop with no intermediate storage” or “flow shop with finite buffers” or “flowshop with finite buffers” or “flow shop with finite intermediate storage”.	60	14
	“flow shop with limited storage” or “flowshop with limited storage”	5	2
	“assembly line with blocking”	1	0
Scopus	“flow shop with blocking” or “flowshop with blocking” or “blocking flow shop” or “blocking flowshop” and not “flexible flow shop” and not “parallel flow shop”.	221	27
	“flow shop with limited buffers” or “flowshop with limited buffers” or “flow shop with no intermediate storage” or “flowshop with no intermediate storage” or “flow shop with finite buffers” or “flowshop with finite buffers” and not “flexible flow shop” and not “flexible flowshop” and not “hybrid flow shop” and not “hybrid flowshop” and not “distributed flowshop” and not “distributed flowshop”.	64	2
	“flow shop with limited storage” or “flowshop with limited storage” and not “hybrid flowshop” and not “hybrid flow shop” and not “flexible flowshop” and not “flexible flow shop”.	40	3
	“assembly line with blocking”	3	1
		Total	139

Fonte: Autor, 2019.

Para remover resultados redundantes de ambas as bases de dados, a busca foi realizada na seguinte ordem: primeiro, as palavras-chave mostradas na Tabela 1 foram utilizadas na base *ISI Web of Knowledge* e, com os resultados filtrados em mãos, todas as referências repetidas no *Scopus* foram excluídas. Do total encontrado, 139 artigos foram filtrados. As seções seguintes são estruturadas de acordo com o grupo de medidas similares, seguindo o modelo proposto por Rossit *et al.* (2018): 1) funções-objetivo baseada em tempo de conclusão, que inclui duração total da programação (C_{max}) e tempo total de fluxo ($\sum_i C_i$); 2) funções-objetivo baseada em datas de entrega; 3) Tempo de ciclo; 4) Funções multi-objetivo e; 5) funções-objetivo baseadas

em custos estocásticos. As Tabelas 2, 3, 4, 5, 6 e 7 seguem, de forma análoga, a sumarização do resultados feitas por Ruiz e Vázquez-Rodríguez (2010) e Rossit *et al.* (2018), onde as publicações são dispostas de acordo com a notação do problema e comentários, em base temporal. A maioria das referências propôs métodos de solução, isto é, métodos exatos e métodos heurísticos. Os métodos exatos são divididos em programação linear inteira mista (MILP), *branch & bound* e programação dinâmica. Os métodos heurísticos foram subdivididos heurísticas construtivas e metaheurísticas e buscas locais. Uma terceira porcentagem de artigos considerou a análise dos problemas e revisão de literatura e está incluída nas respectivas subseções, com exceção de Hall e Sriskandarajah (1996) e Emmons e Vairaktarakis (2013).

2.3 FUNÇÕES-OBJETIVO BASEADAS EM DATA DE TÉRMINO

2.3.1 Duração total da programação (C_{max})

A importância da duração total da programação (C_{max}) como medida de desempenho para o problema *flow shop* com bloqueio é refletida no número de publicações que o adotaram. Dos 139 artigos abordados, C_{max} é a função mono-objetivo adotada por 62% do total. O objetivo dos artigos incluiu a proposição de métodos de solução e análise de complexidade de vários tipos de ocorrências de bloqueio. A Tabela 2 mostra uma sumarização dos artigos cobertos que adotaram C_{max} .

Tabela 2 - Publicações encontradas para o problema *flow shop* com minimização da C_{max} (Continua...)

Referências	Problemas	Comentários
Levner (1969)	$F_m block C_{max}$	B&B
Reddi; Ramamoorthy (1972)	$F_2 block C_{max}$	O problema pode ser reduzido ao problema do caixeiro viajante (PCV). O problema pode ser solucionado em tempo polinomial pelo algoritmo de Gilmore e Gomory.
Dutta; Cunningham (1975)	$F_2 b_{j,j+1} C_{max}$	Programação dinâmica
Papadimitriou; Kanellakis (1980)	$F_2 b_{j,j+1} C_{max}$	O problema é NP-Completo.

(Continuação)

Referências	Problemas	Comentários
Suhami; Mah (1981)	$F_m block C_{max}$	B&B
Leisten (1990)	$F_m b_{jj+1} C_{max}$	Heurísticas construtivas
Logendran; Sriskandarajah (1993)	$F_2 block, st_{ant} C_{max}$	O problema é fortemente NP-difícil. A heurística proposta usa o algoritmo de Gilmore e Gomory para solucionar o problema.
Levner et al. (1995)	$F_2 block, st_{ant} C_{max}$	O problema da célula robótica com um robô e tempos de <i>setup</i> para minimização de C_{max} pode ser solucionado em tempo polinomial pelo algoritmo de Gilmore e Gomory.
Norman (1999)	$F_m b_{jj+1}, st_{sd} C_{max}$	TS
Nowicki (1999)	$F_m b_{jj+1} C_{max}$	TS
Grabowski; Pempera (2000)	$F_m wB, no-wait, block C_{max}$	TS
Caraffa et al. (2001)	$F_m block C_{max}$	GA
Ronconi (2004)	$F_m block C_{max}$	Heurística construtiva
Li; Tang (2005)	$F_m b_{jj+1} C_{max}$	TS
Ronconi (2005)	$F_m block C_{max}$	B&B
Soukhal et al. (2005)	$F_2 \rightarrow D block, v, c_i, t_k C_{max}$	$F_2 \rightarrow D block, v=1, c_i=2 C_{max}$, $F_2 \rightarrow D block, v=1, c_i=2, t_j=h C_{max}$ e $F_2 \rightarrow D block, v=1, c_i=3 C_{max}$ são fortemente NP-difíceis.
Martinez et al. (2006)	$F_m block, RCB C_{max}$	$F_2 RCB C_{max}$ é equivalente a $1 C_{max}$. $F_3 RCB C_{max}$ is um problema polinomial $F_4 RCb(1,2), block(2,3), RCB(3,4) C_{max}$, $F_3 RCb(1,2), block(2,3) C_{max}$ e $F_3 block(1,2), RCB(2,3) C_{max}$ são problemas polinomiais. $F_5 RCb C_{max}, F_4 RCb(1,2), RCB(2,3), block(3,4) C_{max}$, $F_4 block(1,2), RCB(2,3), RCB(3,4) C_{max}$ e $F_4 block(1,2), RCB(2,3), block(3,4) C_{max}$ são problemas NP-difíceis.
Wang et al. (2006)	$F_m b_{jj+1} C_{max}$	GA
Companys; Mateo (2007)	$F_m block C_{max}$	B&B
Grabowski; Pempera (2007)	$F_m block C_{max}$	TS

(Continua...)

(Continuação)

Referências	Problemas	Comentários
Yuan et al. (2007)	$F_2 \rightarrow D block, v, c_i, t_k C_{max}$	$F_2 \rightarrow D block, v=1, c_i=2 C_{max}$ é um problema binário NP-difícil $F_2 \rightarrow D block, v=1, c_i=3, t_j=h C_{max}$ is fortemente NP-difícil
Liu et al. (2008)	$F_m b_{j+1} C_{max}$	PSO
Pitty; Karimi (2008)	$F_m block C_{max}$ and $F_m block, st_{sd} C_{max}$	MILP
Jarboui et al. (2009)	$F_m block C_{max}$	EDA
Liu; Kozan (2009)	$F_m wB, no-wait, block, b_{j+1} C_{max}$	Heurística construtiva
Qian et al. (2009)	$F_m b_{j+1} C_{max}$	DE
Carlier et al. (2010)	$F_m block, t_j C_{max}$	GA
Companys; Ribas; Mateo (Mane) (2010)	$F_m block C_{max}$	Heurística construtiva
Companys; Ribas; Mateo (Manel) (2010)	$F_m block C_{max}$	SA
Duan et al. (2010)	$F_m block C_{max}$	HS
Gong et al. (2010)	$F_2 no-wait, block, batch, sharedst C_{max}$	Heurística construtiva
Lee et al. (2010)	$F_2 block, ap_{ij} C_{max}$	B&B
Trabelsi et al. (2011)	$F_m RCB, block, RCB^* C_{max}$	$F_4 RCB(1,2), block(2,3), RCB(3,4) C_{max}$ e $F_3 RCB^* C_{max}$ são problemas polinomiais MILP
Wang et al. (2010)	$F_m block C_{max}$	DE
Liang et al. (2011)	$F_m block C_{max}$	PSO
Pan et al. (2011)	$F_m b_{j+1} C_{max}$	HS
Pan et al. (2011)	$F_m b_{j+1} C_{max}$	DE
Ribas et al. (2011)	$F_m block C_{max}$	IG
Tang; Tang (2011)	$F_2 b_{j+1}, batch, st_{si} C_{max}$	PSO
Wang et al. (2011)	$F_m block C_{max}$	HS
Bao et al. (2012)	$F_m block C_{max}$	HS
Bautista et al. (2012)	$F_m block C_{max}$	Programação dinâmica com Limitantes
Davendra et al. (2012)	$F_m block C_{max}$	DE
Han et al. (2012)	$F_m block C_{max}$	ABC
Pan (et al. (2012)	$F_m block C_{max}$	ABC

(Continua...)

(Continuação)

Referências	Problemas	Comentários
Maleki-Daroukolaie et al. (2012)	$3AF block, st_{sd} \alpha(\sum_i W_i C_i / W) + (\alpha-1)C_{max}$	SA
Pan; Wang (2012)	$F_m block C_{max}$	Heurística construtiva
Sauer; Sauvey (2012)	$F_m RCb C_{max}$	GA
Trabelsi et al. (2012)	$F_m Wb, block, RSb, RCb^*, RCb C_{max}$	GA
Wang et al. (2012)	$F_m block C_{max}$	SA
Wang; Tang (2012)	$F_m block C_{max}$	PSO
Chowdhury et al. (2013)	$F_m block C_{max}$	GA
Davendra; Bialic-Davendra (2013)	$F_m block C_{max}$	SOMA
Davendra et al. (2013)	$F_m block C_{max}$	SOMA
Lin; Ying (2013)	$F_m block C_{max}$	AIS
Pan et al. (2013)	$F_m block C_{max}$	MMA
Ribas; Companys (2013)	$F_m block C_{max}$	VNS
Chen et al. (2014)	$F_2 block, batch, r_{ij} C_{max}$	DE
Davendra et al. (2014)	$F_m block C_{max}$	SSA
Moslehi; Khorasanian (2014)	$F_m b_{j+1} C_{max}$	VNS
Zhao et al. (2014)	$F_m b_{j+1} C_{max}$	PSO
Ding et al. (2015)	$F_m block C_{max}$	IG
Han et al. (2015)	$F_m block C_{max}$	ABC
Khorramizadeh; Riahi (2015)	$F_m Wb, block, RSb, RCb^*, RCb C_{max}$	ABC
Liu (Li.; Ren (2015)	$F_m block C_{max}$	VNS-SSA
Sadaqa; Moraga (2015)	$F_m block C_{max}$	Meta-RaPS
Tasgetiren et al. (2015)	$F_m block C_{max}$	DE
Wang et al. (2015)	$F_m block C_{max}$	CuckooSA

(Continua...)

(Conclusão.)

Referências	Problemas	Comentários
Zhang et al. (2015)	$F_m block C_{max}$	VNS-SA
Al-Salem; Kharbeche (2016)	$F_2 batch(1), block, p_{ij}^{min} \leq p_{ij} \leq p_{ij}^{max} C_{max}$	GA
Eddaly et al. (2016)	$F_m block C_{max}$	PSO
Han et al. (2016)	$F_m block C_{max}$	FFOA
Fu et al. (2017)	$F_m block, ap_{ij} C_{max}$	CRO-VNS
Khorasanian; Moslehi (2017)	$F_2 block, mtflx, prmp C_{max}$	VNS
Riahi et al. (2017)	$F_m Wb, block, RSb, RCb^*, RCb C_{max}$	SSA
Takano; Nagano (2017)	$F_m block, st_{sd} C_{max}$	B&B
Tasgetiren et al. (2017)	$F_m block C_{max}$	IG
Toumi et al. (2017)	$F_m block C_{max}$	B&B
Koulamas; Panwalkar (2018)	$F_m block, ord C_{max}$	Regra de sequenciamento
Liu; Wang; Zhang (2018)	$F_m block C_{max}$	IBBO
Panwalkar; Koulamas (2018)	$F_m block C_{max}$	Novo forma de representar <i>flow shop</i> com bloqueio graficamente
Riahi et al. (2018)	$F_m st_{sd}, RCb, RSb C_{max}$	CGLS
Shao (Pi.; Shao (2018)	$F_m block, st_{sd} C_{max}$	DWWO
Shao; Pi; Shao (2018)	$F_m block C_{max}$	EDA-VNS
Ozolins (2019)	$F_m block C_{max}$	Programação dinâmica com limitantes
Takano; Nagano (2019)	$F_m block, st_{sd} C_{max}$	Heurística construtiva

Abreviaturas: B&B – *Branch & Bound*; TS – *Tabu Search*; GA – *Genetic Algorithm*; PSO – *Particle Swarm Optimization*; DE – *Differential Evolution*; GRASP – *Greedy Randomized Adaptive Search Procedure*; HS – *Harmony Search*; IG – *Iterated Greedy*; ABC – *Artificial Bee Colony*; SA – *Simulated Annealing*; SOMA – *Self-organising Migrating Algorithm*; AIS – *Artificial Immune System*; MMA – *Memetic Algorithm*; ILS – *Iterated Local Search*; VNS – *Variable Search Neighborhood*; CuckooSA – *Cuckoo Search Algorithm*; MDGSO – *Multi-objective discrete group search optimizer*; FFOA – *Fruit Fly Optimization Algorithm*; VBIH; CRO-VNS – *Chemical Reaction Optimization with VNS*; CSA – *Clustering Search Algorithm*; SSA – *Scatter Search Algorithm*; IBBO – *Improved Biography-Based Optimization Algorithm*; CGLS – *Constraint Guided Local Search*; DWWO – *Discrete Water Wave Optimization*; EDA-VNS – *Estimation Distribution Algorithm with VNS*; Meta-RaPS – *Metaheuristic for Random Priority Search*.

Fonte: Autor, 2019.

Devido à diversidade de problemas encontrados, nesta subseção, as publicações também são subcategorizadas de acordo com as restrições adotadas. Isso facilitará o leitor a perceber a contextualização histórica do problema, desde a primeira até a última publicação encontrada até o momento. Assim, esta seção é dividida em quatro subseções, que representam os seguintes problemas: $F_m|block|C_{max}$, $F_m|block,\beta_i|C_{max}$ and $F_m|RCB,block\text{ or/and }RCb^*,\beta_i|C_{max}$ and $F_m|b_{j,j+1},\beta_i|C_{max}$.

2.3.1.1 *Flow shop* com bloqueio e minimização de C_{max}

Levner (1969) propôs um *branch & bound* (B&B) para o problema $F_m|block|C_{max}$. Os limitantes inferiores foram derivados de propriedades do bloqueio. Reddi e Ramamoorthy (1972) demonstraram que o problema $F_2|block|C_{max}$ pode ser reduzido ao problema do caixeiro viajante e propuseram o algoritmo de Gilmore e Gomory (GILMORE; GOMORY, 1964) como método de solução, que resolve o problema em tempo polinomial.

Suhami e Mah (1981) propuseram um B&B para o problema $F_m|block|C_{max}$. Os autores desenvolveram limitantes inferiores baseados em estimativas da média e da variância de C_{max} parcial de sequências parciais. Um exemplo numérico do B&B proposto e alguns resultados computacionais são mostrados.

Caraffa et al. (2001) propuseram um Algoritmo Genético (GA) para o problema $F_m|block|C_{max}$. GA é uma metaheurística que imita o processo de seleção natural das espécies. GA inicia-se a partir de uma população inicial, onde cada indivíduo representa uma solução do problema. Em cada geração, o valor *fitness* de cada indivíduo é avaliado. Múltiplos indivíduos são selecionados por um operador de seleção e os indivíduos selecionados são modificados usando operadores de cruzamento e de mutação para formar uma nova população melhorada. O algoritmo termina quando o critério de parada é atingido. O GA proposto pelos autores gera a população inicial aleatoriamente e usa o operador *roulette section wheel* na fase de seleção e os operadores *two-point crossover* e troca entre pares de tarefas nas fases de cruzamento e mutação, respectivamente. Resultados computacionais mostraram que o proposto GA superou a heurística GENIUS em qualidade de solução e em tempo de computacional.

Ronconi (2004) propôs a heurística MinMax (MM) para o problema $F_m|block|C_{max}$. MM é semelhante à heurística PF, desenvolvida por McCormick et al. (1989). Em MM, a

primeira e a última tarefa são escolhidas de acordo com o tempo de processamento mais curto na primeira e na última máquina, respectivamente. A partir da segunda posição, o algoritmo considera as somas dos valores máximos entre os tempos de processamento de máquinas consecutivas e os tempos totais de processamento da tarefa. O mecanismo de inserção de NEH é aplicado às soluções geradas pelas heurísticas de MM e Profile Fitting (PF), criando MME e PFE. NEH é uma heurística construtiva proposta por Nawaz et al. (1983) para o problema $F_m|pmu|C_{max}$. NEH é composto por duas fases. Na primeira fase, as tarefas são ordenadas em ordem não crescente do tempo total de processamento, gerando uma Lista Inicial (LI). Na segunda fase, as duas primeiras tarefas da LI são removidas e inseridas na sequência parcial S . A sequência parcial entre as duas combinações possíveis das duas tarefas que gera o menor C_{max} é escolhida para a próxima iteração. A próxima tarefa da LI é então inserida em todas as posições possíveis da sequência parcial. A sequência parcial que gerar o menor C_{max} é escolhida. O processo é repetido até que todas as tarefas da LI sejam programadas. Ao longo da Seção 2, a segunda fase de NEH será referida como mecanismo de inserção NEH. Resultados computacionais mostraram que, em média, PF alcançou melhores resultados que MM. Em relação ao número de melhores soluções, PFE e MME apresentaram desempenho superior em comparação a NEH.

Ronconi (2005) desenvolveu um B&B para o problema $F_m|block|C_{max}$. A autora propôs um limitante inferior (LB_{RON}) para C_{max} que leva em conta a estimativa dos tempos de partida do conjunto de tarefas ainda não programadas. Resultados computacionais mostraram que o limite inferior proposto obteve melhor desempenho que o limite inferior proposto por Ronconi e Armentano (2001).

Abadi (2007) estenderam a técnica de desaceleração das operações propostas por Abadi et al. (2000) para o cálculo da função-objetivo do problema $F_m|block|C_{max}$.

Company e Mateo (2007) propuseram o algoritmo LOMnicki PENDular (LOMPEN) para o problema $F_m|block|C_{max}$. LOMPEN aplica o algoritmo B&B simultaneamente na instância do problema direta e inversa. Um limitante inferior inspirado no de Ronconi e Armentano (2001) é adaptado para aplicação de LOMPEN ao problema $F_m|block|C_{max}$. Os autores encontraram novas melhores soluções para os problemas da instância de (TAILLARD, 1993) para até 200 tarefas e 10 máquinas.

Grabowski e Pempera (2007) propuseram duas versões do algoritmo *Tabu Search* (TS) para o problema $F_m|block|C_{max}$. A ideia básica da TS consiste em partir de uma solução inicial e procurar, em sua vizinhança, uma solução que minimiza a função-objetivo. Caso haja uma solução melhor, esta se torna a solução inicial na próxima iteração. Uma lista tabu, que

armazena movimentos já feitos, é usada para evitar que a busca fique estagnada em um mínimo local, bem como também para orientar a busca para regiões promissoras. Esta lista registra, por um determinado número de iterações, os movimentos realizados, tratando-os como proibitivos em movimentos futuros. O algoritmo termina quando um critério de parada é alcançado. TS e TS+M aplicam as propriedades de *block* e *anti-block*, desenvolvidas por Grabowski e Pempera (2000) para remoção de “movimentos inúteis”. Adicionalmente, TS+M usa a busca local de multimovimentos, que aplica vários movimentos potenciais simultaneamente a uma sequência candidata. Resultados computacionais mostraram que ambas as heurísticas superaram GA de Caraffa et al. (2001) e B&B de Ronconi (2005).

Pitty e Karimi (2008) propuseram 17 modelos de programação linear inteira mista (MILP) para os problemas $F_m|block|C_{max}$ e $F_m|block,st_{sd}|C_{max}$. Os autores realizaram uma extensa avaliação computacional com MILP's da literatura.

Jarboui et al. (2009) propuseram uma versão do algoritmo *Estimation of Distribution Algorithm* (EDA) para o problema $F_m|block|C_{max}$. EDA é um algoritmo evolucionário baseado em populações que explora o espaço de soluções por amostragem de um modelo probabilístico, atualizado de acordo com as melhores soluções promissoras até a dada iteração. EDA inicia com uma população inicial de soluções candidatas. Cada indivíduo da população é ranqueada de acordo com sua função *fitness*. Um subconjunto com as soluções mais promissoras é selecionado por um operador de seleção. Então, os parâmetros do modelo de probabilidade são atualizados de acordo com as melhores soluções encontradas e novas soluções são geradas por amostragem da distribuição codificada pelo modelo. O algoritmo repete esse procedimento até que o critério de parada seja atendido. A metaheurística proposta, denominada H-EDA, inicia-se com uma população inicial gerada pelo método NEH e aleatoriamente. Como critério de seleção, as Q soluções são selecionadas de acordo com C_{max} . Um modelo de distribuição probabilística é desenvolvido com base na posição das tarefas na sequência. O pior indivíduo é substituído pelo novo indivíduo de acordo com C_{max} . Uma busca local é aplicada aos novos indivíduos, dado um valor de probabilidade de melhoria. Resultados computacionais mostraram que H-EDA apresentou desempenho superior a B&B de Ronconi (2005) e TS+M de Grabowski e Pempera (2007).

Company's, Ribas e Mateo (2010) propuseram heurísticas construtivas inspiradas em NEH para o problema $F_m|block|C_{max}$. Os autores utilizaram soluções geradas pelas heurísticas PF (MCCORMICK et al., 1989), Trapezium (COMPANY'S, 1966), PO (POUR, 2001) e N&M (NAGANO; MOCCELLIN, 2002) como sequências iniciais. Na segunda etapa de NEH, um novo mecanismo de desempate é proposto para avaliar sequências parciais que obtem C_{max}

iguais. Resultados computacionais indicaram PLE2, que consiste na aplicação da heurística PF para instância do problema inversa, como melhor método.

Companys, Ribas e Mateo (2010) propuseram heurísticas melhorativas para o problema $F_m|block|C_{max}$. Os autores usaram as heurísticas de NEH, POUR e N&M como soluções iniciais para uma variação do algoritmo *Non-Exhaustive Descending Search* (NEDA), denominado *Soft Simulated Annealing* (SSA). NEDA tenta melhorar a solução trocando duas tarefas em quaisquer duas posições da sequência. Resultados experimentais mostraram que os procedimentos desenvolvidos melhoraram consideravelmente as soluções iniciais das heurísticas consideradas.

Duan et al. (2010) propuseram uma versão híbrida do algoritmo *Harmony Search* (HS) para o problema $F_m|block|C_{max}$. HS é um algoritmo estocástico de população que tenta simular o processo natural de performance musical quando um músico procura por um melhor estado de harmonia. Cada permutação é representada por um vetor n -dimensional de harmonia. O HS requer uma população inicial de vetores de harmonia que são armazenados em uma memória de harmonia. Durante a fase de improvisação, novos indivíduos são gerados usando consideração de memória, ajuste do tom e uma seleção aleatória. A memória de harmonia é atualizada comparando o novo candidato com a pior solução da consideração de memória. O processo continua até que um critério de parada seja atendido. A metaheurística proposta HHS utiliza uma versão modificada de NEH e o método aleatório para gerar a população inicial. Na fase de improvisação, uma nova solução é gerada de acordo com a consideração da memória ou é selecionada aleatoriamente. Se for gerado pela consideração de memória, um ajuste do tom (*pitch adjustment*) é aplicado ao novo indivíduo. Uma busca local de inserção é aplicada a cada novo indivíduo gerado. Resultados experimentais mostraram que HHS superou GA Caraffa et al. (2001) e TS e TS+M Grabowski e Pempera (2007).

Wang et al. (2010) propuseram algumas versões do algoritmo *Differential Evolution* (DE) para o problema $F_m|block|C_{max}$. DE é um algoritmo estocástico baseado em população, que usa as informações diferenciais entre os indivíduos para conduzir a otimização global. O algoritmo parte de uma população-alvo e a cada geração adota um operador de mutação para gerar uma população mutante, um operador de cruzamento para gerar indivíduos experimentais e um operador guloso para determinar uma nova população-alvo para a próxima geração. Os autores propuseram uma versão adaptada do algoritmo, chamada *Discrete DE* (DDE), para o problema de otimização discreta. A segunda versão, denominada *Hybrid Discrete DE* (HDDE), incorpora uma busca local, que é realizada com uma certa probabilidade após a fase de cruzamento. Resultados experimentais mostraram que as metaheurísticas propostas superaram

HDDE proposta por Qian et al. (2009), GA de Caraffa et al. (2001), TS e TS+M de Grabowski e Pempera (2007).

Liang et al. (2011) desenvolveram uma versão do algoritmo *Particle Swarm Optimization* (PSO) para o problema $F_m|block|C_{max}$. O princípio básico de PSO é simular o comportamento social da revoada de aves ou da ordem dos peixes, bem como os meios de troca de informações entre eles. Um enxame consiste de n indivíduos, chamados de partículas, que voam em um espaço de busca de dimensão n . Em *scheduling*, cada partícula representa uma sequência de tarefas. Cada partícula é definida por vetores com n dimensões com as seguintes características: a) a posição da i -ésima partícula no espaço de busca; b) a velocidade com que a partícula i se move; c) a melhor posição da partícula i ; e d) a melhor partícula global do enxame até a t -ésima iteração. A posição e a velocidade da i -ésima partícula é ajustada em cada iteração t , que é guiada pela melhor posição visitada por si mesma e pela melhor posição da melhor partícula do enxame. O processo continua até que um critério de parada seja alcançado. Afim de garantir um enxame com nível de qualidade e diversidade, o proposto DMS-PSO usa uma variante de NEH para gerar os indivíduos iniciais. Então, o algoritmo divide o enxame em sub-enxames, no intuito de desacelerar a velocidade de convergência da população e aumentar a diversidade. Um PSO local é aplicado a cada sub-enxame. A cada R gerações, os sub-enxames são aleatoriamente reagrupados. A fim de melhor atender o *trade-off* entre diversidade e velocidade de convergência, uma busca local é aplicada aos melhores indivíduos de cada sub-enxame. Resultados experimentais mostraram que a metaheurística proposta superou HDDE de Qian et al. (2009) e TS e TS+M de Grabowski e Pempera (2007).

Ribas et al. (2011) desenvolveram duas versões do algoritmo *Iterated Greedy* (IG), IG1 e IG2, para o problema $F_m|block|C_{max}$. IG inicia-se a partir de uma solução inicial e a cada iteração uma solução candidata parcial é obtida removendo d tarefas de uma solução candidata completa. Em seguida, a sequência é reconstruída a partir da solução candidata parcial. Um critério de aceitação decide se a nova solução se tornará a nova solução incumbente. O processo é repetido até que o critério de parada seja alcançado. Nas duas versões de IG propostos, a heurística NEH é usada para gerar a solução inicial. Durante a fase de reconstrução, os autores propuseram alguns mecanismos de desempate para avaliar sequências parciais com C_{max} iguais. Uma fase de melhoria é incorporada ao algoritmo, usando NEDA. A fim de evitar explorar a vizinhança sempre na mesma ordem, os autores desenvolveram uma ferramenta chamada revólver, que é usada para selecionar duas posições da sequência aleatoriamente durante a aplicação de NEDA. A diferença entre os dois métodos propostos é que IG2 usa a ferramenta

revólver na fase de melhoria e IG1 não. Resultados computacionais mostraram que tanto o IG1 quanto o IG2 superaram o HDDE de Wang et al. (2010).

Wang et al. (2011) propuseram três versões de HS para o problema $F_m|block|C_{max}$. A primeira versão proposta de HS é chamada HS padrão e usa NEH-WPT para gerar uma solução da população inicial. NEH-WPT cria a Lista Inicial (LI) ordenando as tarefas de acordo com os valores não decrescentes do tempo total de processamento e aplica o mecanismo de inserção NEH à lista inicial. A segunda versão de HS, denominada *best modified global* HS (mgHS), usa uma nova regra de ajuste do tom. A terceira versão, chamada *Hybrid* mgHS (hmgHS), adiciona uma busca local de inserção a mgHS. Resultados computacionais mostraram que todos os algoritmos propostos superaram HGA de Wang et al. (2006), TS+M de Grabowski e Pempera (2007), HPSO de Liu et al. (2007) e IG de Ribas et al. (2011).

Davendra et al. (2012) desenvolveu uma versão clusterizada da metaheurística DE, denominada EDEC para o problema $F_m|block|C_{max}$. EDEC usa o princípio de clusterização para substituir os velhos indivíduos da população por novos indivíduos gerados. Além das etapas padrão do DE, EDEC também aplica uma busca local para explorar a vizinhança da solução candidata. Por meio de testes computacionais, os autores constataram que EDEC superou GA de Caraffa et al. (2001), TS e TS+M de Grabowski e Pempera (2007), HDDE de Qian et al. (2009) e HDDE de Wang et al. (2010).

Bao et al. (2012) propuseram uma versão da metaheurística HS, denominada *Improved* HS (IHS) para o problema $F_m|block|C_{max}$. IHS combina o algoritmo HS com o algoritmo GA e uma busca local de inserção para melhorar as soluções. Resultados computacionais mostraram que a metaheurística proposta supera HS, GA e Hybrid HS + GA de Bao et al. (2011).

Bautista et al. (2012) propuseram uma programação dinâmica com limitantes (BDP) para o problema $F_m|block|C_{max}$. BDP combina recursos de programação dinâmica e B&B. Os autores adaptaram os limitantes inferiores propostos por Lageweg et al. (1978), como limitantes parciais generalizados, considerando que o problema $F_m|pmu|C_{max}$ é uma relaxação do problema $F_m|block|C_{max}$. O BDP proposto foi comparado com os melhores resultados para os problemas das instâncias de Taillard, fornecidos por Ribas et al. (2011).

Han et al. (2012) propuseram uma versão da metaheurística *Artificial Bee Colony* (ABC) para o problema $F_m|block|C_{max}$. ABC é um algoritmo baseado em inteligência de enxame que simula o comportamento de forrageamento das abelhas campeiras, observadoras e exploradoras. ABC requer uma população de fontes de alimento, onde cada fonte de alimento corresponde a uma sequência de tarefas. Então, essas fontes de alimento são atualizadas por três tipos de abelhas. Primeiramente, uma abelha campeira gera uma nova fonte de alimento

baseada na atual. A abelha observadora, modifica a fonte de alimento compartilhada pela abelha campeira. Uma nova fonte de alimento é produzida para atualizar as soluções inalteradas usando uma abelha exploradora. A função *fitness* da fonte de alimento é calculada em todas as fases. A metaheurística proposta IABC usa MM, NEH, NEH-WPT, uma versão modificada do MM e o método aleatório para gerar a população inicial. No estágio de abelha empregada, quatro operadores baseados em inserção e troca entre pares de tarefas são aplicados para gerar novas soluções. Na fase da abelha observadora, uma busca local de inserção, desenvolvida por Ruiz e Stützle (2008) é aplicada para tentar melhorar as soluções. Por fim, o algoritmo DE de Wang et al. (2010) é aplicado no estágio das abelhas exploradoras, para gerar novas soluções. O IABC proposto obteve melhores resultados que GA de Caraffa et al. (2001), TS e TS+M de Grabowski e Pempera (2007), HDDE de Wang et al. (2010) e Discrete ABC de Han et al. (2011).

Pan e Wang (2012) desenvolveram algumas heurísticas construtivas baseadas na combinação de PF, NEH e *Referenced Local Search* (RLS) para o problema $F_m|block|C_{max}$. Os autores desenvolveram wPF e PW. Ambas as heurísticas usam os mesmos passos de PF, diferenciando-se apenas da função de avaliação para selecionar a próxima tarefa da sequência parcial. Para melhorar a qualidade da solução, o mecanismo de inserção NEH é adicionado à PF, wPF e PW, criando PF-NEH(x), wPF-NEH(x) e PW-NEH(x), onde x é o número de iterações que as heurísticas são executadas. Por fim, PF-NEH_{LS}(x), wPF-NEH_{LS}(x) e PW-NEH_{LS}(x) são propostas aplicando RLS é aplicada à solução gerada por PF-NEH(x), wPF-NEH(x) e PW-NEH(x), respectivamente. Resultados experimentais mostraram que PF-NEH_{LS}(x), wPF-NEH_{LS}(x) e PW-NEH_{LS}(x) obtiveram os melhores resultados.

Wang e Tang (2012) desenvolveram *Discrete PSO* (DPSO) para o problema $F_m|block|C_{max}$. DPSO usa uma perturbação auto-adaptativa para evitar a convergência prematura das soluções. A metaheurística proposta também aplica o algoritmo *Variable Search Neighborhood* (VNS), inspirada em Mladenovic e Hansen (1997), a melhor partícula global encontrada. Resultados computacionais mostraram que DPSO superou TS+M de Grabowski e Pempera (2007) e os algoritmos DPSO de Rameshkumar, Suresh e Mohanasundaram (2005) e de Ponnambalam et al. (2009).

Wang et al. (2012) propuseram *Three Phase Algorithm* (TPA) para o problema $F_m|block|C_{max}$. TPA é composto por NEH seguida por uma versão de *Simulated Annealing* (SA). Na primeira fase de TPA, as tarefas são ordenadas pela ordem não decrescente da soma ponderada dos tempos médios de processamento das tarefas e o desvio padrão dos tempos de processamento. Na segunda fase, o mecanismo de inserção NEH é aplicado à sequência inicial gerada na fase 1. Na terceira fase, é aplicado um *Simulated Annealing* modificado para melhoria

da solução. Resultados computacionais mostraram que o TPA superou o HDDE de Wang et al. (2010) e IG1 e IG2 de Ribas et al. (2011) em qualidade da solução.

Chowdhury et al. (2013) propuseram um versão de GA denominada *Novel GA* (NGA) para o problema do caixeiro viajante (TSP) e o problema $F_m|block|C_{max}$. NGA aplica uma mutação forçada às piores sequências da população. Um mecanismo de mutação multinível composto por mais de um operador (troca e inversão) é usado durante a fase de mutação. Para garantir um fluxo de qualidade bidirecional, NGA usa o operador de *crossover OXXI*. Resultados experimentais mostraram que a metaheurística proposta superou o B&B de Ronconi (2005), TS+M de Grabowski e Pempera (2007), HDDE de Wang et al. (2010).

Davendra e Bialic-Davendra (2013) propuseram uma versão do Algoritmo *Discrete Self-Organising Migrating Algorithm* (DSOMA) para o problema $F_m|block|C_{max}$. SOMA é um algoritmo evolutivo populacional baseado no comportamento competitivo e cooperativo de criaturas inteligentes para resolver um problema comum. SOMA inicia-se a partir de uma população inicial, onde cada indivíduo corresponde a uma solução. Uma fase de migração é aplicada, onde os indivíduos são modificados por movimentos em direção à melhor solução da população. Os movimentos consistem em saltos, determinados por um parâmetro de salto até que o indivíduo atinja a posição final de um parâmetro baseado em *path relinking*. Em DSOMA, a população inicial é gerada aleatoriamente por um método de inserção. Para criar novos indivíduos, o parâmetro de saltos mínimos que cada indivíduo circunavega é usado para gerar novos indivíduos. Uma busca local 2-OPT é aplicada ao melhor novo indivíduo gerado. Resultados experimentais mostraram que o DSOMA superou os resultados gerados por B&B de Ronconi (2005), TS+M de Grabowski e Pempera (2007), HDDE de Wang et al. (2010).

Davendra et al. (2013) propuseram um algoritmo DSOMA para o problema $F_m|block|C_{max}$. A técnica de *lozzi map* é incorporada ao DSOMA, para entrar no lugar do gerador de números pseudo-aleatórios usado para resolver o problema. Resultados experimentais mostraram que DSOMA obteve melhor desempenho que TS+M de Grabowski e Pempera (2007) e HDDE de Qian et al. (2009).

Lin e Ying (2013) propuseram uma versão do algoritmo *Artificial Immune System* (AIS) com SA incorporado e um *Pure AIS* (PAIS) para o problema $F_m|block|C_{max}$. O AIS é um algoritmo estocástico inspirado em sistemas imunológicos biológicos (BIS). No AIS, cada anticorpo de uma população representa uma solução. Um valor de afinidade está associado a cada anticorpo. De acordo com uma função de afinidade predefinida, é calculado um número de clones, gerados a partir de cada anticorpo. Os clones são mutados para gerar novos anticorpos. Os novos anticorpos são avaliados novamente, e os piores indivíduos da população

são substituídos pelos melhores clones mutados. O processo é repetido até que um critério de término seja atendido. Na metaheurística RAIS, os anticorpos são gerados aleatoriamente para compor a população inicial e um determinado número de anticorpos está sujeito a ser clonado. Todos os clones são mutados usando o operador de troca entre pares de tarefas em posições aleatórias. Após a atualização da população, um determinado número de anticorpos é mutado novamente e as novas soluções que geraram os piores valores de afinidade podem substituir o indivíduo original sob uma função de probabilidade baseada no mecanismo de aceitação de SA. A metaheurística foi comparada contra o B&B proposto por Ronconi (2005). Os resultados computacionais mostram que ambas as metaheurísticas superaram TS+M de Grabowski e Pempera (2007), HDDE de Wang et al. (2010).

Pan et al. (2013) propuseram uma versão modificada do Algoritmo Memético (MMA) para o problema $F_m|block|C_{max}$. MMA é uma metaheurística baseada em população, inspirada na seleção natural dos princípios darwinianos e Dwakins, definida como um processo de evolução cultural capaz de refinamentos individuais. MA repete a avaliação, seleção, cruzamento, mutação e refinamento local, até que um determinado critério de parada seja atendido. Para compor a população inicial, os autores propuseram PF+NEH, que utiliza PF para programar $n - \lambda$ tarefas e NEH para as λ tarefas restantes. Nas fases de seleção, cruzamento e mutação, MMA usa os operadores *tournament selection*, *path relinking* e inserção, respectivamente. Como refinamento local, uma busca local, denominada *Referenced Local Search* (RLS) é aplicada aos novos indivíduos gerados após a mutação. Resultados experimentais mostram que o MMA superou HDDE de Wang et al. (2010), IG de Ribas et al. (2011), HGA de Wang et al. (2006), HPSO de Liu et al. (2008), DPSO de Wang e Tang (2012) e GA's de Tseng e Lin (2010).

Ribas e Companys (2013) propuseram duas versões do algoritmo *Variable Neighborhood Search* (VNS) para o problema $F_m|block|C_{max}$. A principal estratégia do VNS é a mudança sistemática na estrutura da vizinhança dentro de uma busca local, no intuito de escapar de um ótimo local. VNS inicia a partir de uma solução inicial e, enquanto o critério de parada não é atingido, buscas locais são aplicadas à solução. As propostas Paralell VNS (PVNS) e Series VNS (SVNS) usam o procedimento de seleção e o mecanismo de perturbação iguais, mas diferem na fase de melhoria. PW/PW2 de Pan e Wang (2012) são aplicadas respectivamente às instâncias diretas e invertidas para gerar a solução inicial. A perturbação é feita de acordo com os procedimentos de destruição/construção propostos por Ruiz e Stützle (2007). Em PVNS, uma das buscas locais é aplicada, dada uma porcentagem da seleção. Em

SVNS, as buscas locais são aplicadas em série. Resultados experimentais mostraram que PVNS e SVNS superaram HDDE de Wang et al. (2010) e IG de Ribas et al. (2011).

Davendra et al. (2014) propuseram uma versão da metaheurística *Scatter Search Algorithm* (SSA) para o problema $F_m|block|C_{max}$. SSA é uma metaheurística evolucionária que, primeiro cria um conjunto de referência R de soluções e, então, combina de maneira inteligente as soluções entre si, a fim de alcançar melhores soluções. Em cada iteração, as soluções combinadas estão sujeitas a métodos de melhoria e o conjunto R é atualizado. O processo é repetido até que o critério de parada seja alcançado. Os autores incorporaram o operador de cruzamento *two-point crossover* para combinar diferentes soluções e incorporaram uma busca local 2-OPT como método de melhoria. Para conduzir a direção de SSA, os autores usaram a técnica de *chaos map* como geradores de números pseudo-aleatórios. Resultados experimentais evidenciaram que SSA superou HDDE de QIAN et al. (2009), GA de Caraffa et al. (2001) e TS+M de Grabowski e Pempera (2007).

Ding et al. (2015) propuseram um IG com propriedades de bloqueio (B-IG) para o problema $F_m|block|C_{max}$. Baseados em Grabowski e Pempera (2007), os autores desenvolveram algumas novas propriedades para eliminar “movimentos inúteis” de blocos de tarefas. B-IG proposto usa um NEH modificado como solução inicial e aplica as propriedades de bloco propostas durante a fase de reconstrução do IG. Resultados experimentais mostraram que o B-IG superou HDDE de Wang et al. (2010), IG de Ribas et al. (2011), TPA de Wang et al. (2012), RAIS de Lin e Ying (2013) e MMA de Pan et al. (2013).

Han et al. (2015) apresentaram uma versão do algoritmo ABC com DE (ABC-DE) para o problema $F_m|block|C_{max}$. Em ABC-DE, a população inicial é composta por uma solução dada por MME e o restante geradas aleatoriamente. Um DE é incorporado na fase de abelha campeira. O operador *tournament selection* e uma estratégia auto-adaptativa são aplicadas na fase de abelha observadora. Na fase da abelha exploradora, vários operadores de inserção são aplicados às soluções inalteradas. A metaheurística proposta superou TS+M de Grabowski e Pempera (2007), HDDE de Wang et al. (2010), DABC de Han et al. (2011), IABC de Han et al. (2012), DPSO_{SVNS} de Wang e Tang (2012) e EDEC de Davendra et al. (2012).

Liu et al. (2015) propuseram uma metaheurística híbrida, baseada na incorporação de SSA com VNS, para o problema $F_m|block|C_{max}$. Na abordagem híbrida proposta, chamada SS_{VNS}, o banco R é usado para armazenar soluções com boa qualidade e diversidade. Um NEH modificado é aplicado para gerar a solução inicial. Os autores estabeleceram um índice para selecionar cada sequência de R para aplicação da busca local. Uma busca local baseada em blocos de tarefas é usada para gerar novas soluções. O conjunto R é atualizado em cada nova

solução gerada. Resultados experimentais mostraram que SSVNS superou TS+M de Grabowski e Pempera (2007), DMS-PSO de Liang et al. (2011) e alcançou os melhores resultados em 10 de 12 grupos de instâncias em comparação com H-EDA de Jarboui et al. (2009).

Sadaqa e Moraga (2015) propuseram a metaheurística chamada *Metaheuristic for Random Priority Search* (Meta-RaPS) para o problema $F_m|block|C_{max}$. Meta-RaPS cria uma sequência inicial baseada na aleatoriedade seguida por uma etapa de construção que emprega o mecanismo de inserção NEH para construir uma nova solução. Os resultados computacionais mostram que a metaheurística proposta superou mNEH de Wang et al. (2012) em problemas de tamanho pequeno.

Tasgetiren et al. (2015) propuseram uma busca local baseada em população com evolução diferencial (DE_PLS) para o problema $F_m|block|C_{max}$. DE_PLS usa PF-NEH(x), uma versão modificada do GRASP e o método aleatório para gerar a população inicial. O algoritmo inclui um mecanismo que aplica IG ou *Iterated Local Search* ao indivíduo de teste. O novo indivíduo está sujeito a uma RLS, que é guiada pela melhor solução encontrada até o momento. Resultados computacionais mostram que os DE_PLS propostos geram melhores resultados que os algoritmos da literatura testados, exceto para MMA Pan et al. (2013).

Wang et al. (2015) propuseram uma versão do algoritmo *Cuckoo Search Algorithm* (CuckooSA) para o problema $F_m|block|C_{max}$. CuckooSA é um algoritmo de busca estocástica populacional, inspirado no parasitismo de algumas espécies da ave cuco, colocando seus ovos nos ninhos de outras aves hospedeiras. CuckooSA requer uma população inicial de ninhos hospedeiros, onde cada ninho hospedeiro representa uma permutação. Durante as iterações, cada ninho hospedeiro é avaliado pela sua função *fitness* e a população é atualizada de acordo com a qualidade da solução. A população inicial do proposto CuckooSA é composta por soluções geradas por um NEH modificado. Buscas locais de inserção e troca entre pares de tarefas são usadas sequencialmente, dada uma probabilidade, para melhorar a capacidade de exploração. Resultados experimentais mostram que o algoritmo proposto superou DMS-PSO de Liang et al. (2011), TS e TS+M de Grabowski e Pempera (2007), HDDE de Qian et al. (2009), no entanto, com tempo computacional maior.

Zhang et al. (2015) propuseram a hibridização dos algoritmos VNS e SA para o problema $F_m|block|C_{max}$. Cinco estruturas de vizinhança baseadas em troca entre pares de tarefa, inserção e inversão são aplicadas. O HVNSSA usa um método aleatório para gerar uma solução inicial. Um *Simulated Annealing*, que usa o mecanismo de construção/destruição por IG, também é integrado ao HVNS. Resultados computacionais mostraram que HVNSSA é competitivo em comparação com as metaheurísticas avaliadas.

Eddaly et al. (2016) propuseram algumas versões de PSO para o problema $F_m|block|C_{max}$. No proposto PSO Combinatório (CPSO), o enxame inicial é composto por uma partícula gerada por NEH e o restante é gerado aleatoriamente. Novas partículas são geradas de acordo com um procedimento que aplica o operador de troca entre pares de tarefas. No CPSO Híbrido (HCPSO), um ILS é aplicado à melhor solução atual, dado uma probabilidade. Resultados experimentais mostraram que os métodos propostos superaram o GA de Caraffa et al. (2001), TS+M de Grabowski e Pempera (2007), HDDE de Wang et al. (2010).

Han et al. (2016) propuseram o algoritmo denominado *Modified Fruti Fly Optimization* (MFFOA) para o problema $F_m|block|C_{max}$. FFO é uma metaheurística de inteligência de enxame, que simula os comportamentos inteligentes de forrageamento das moscas da fruta. FFO é composto por três etapas: estágio de inicialização, estágio de busca baseado no olfato e estágio de busca por visitação. O MFFOA proposto usa a heurística MME para gerar uma solução enquanto as outras são geradas aleatoriamente para compor a população inicial. No estágio de busca pelo olfato, uma estratégia de busca de vizinhança, baseada em operadores de inserção e troca para frente e para trás, é aplicada às soluções atuais. Em seguida, uma busca local de inserção é aplicada às novas soluções com uma certa probabilidade. Na busca por visitação, a pior solução da população é substituída pela melhor solução encontrada até o momento. Resultados experimentais mostraram que o MFFOA modificado superou HDDE de Wang et al. (2010), IG de Ribas et al. (2011), EDEc de Davendra et al. (2012), DABC de Han et al. (2011), IABC de Han et al. (2012) e DE-ABC de Han et al. (2015).

Tasgetiren et al. (2017) propuseram alguns métodos heurísticos para o problema $F_m|block|C_{max}$. O proposto PFT-NEH(x) é semelhante ao PF-NEH(x). Em PFT, uma lista inicial LI , baseada no *front delay* e na data de término da primeira tarefa, é estabelecida. Na fase iterativa do PFT, a tarefa que ocupa a posição 1 da LI é atribuída à primeira posição da nova sequência. As tarefas restantes são programadas de maneira similar a PF. No total, x sequências são geradas por PFT, onde x é definido de acordo como o número de tarefas. PFT-NEH(x), o mecanismo de inserção NEH é aplicado para as δ últimas tarefas da sequência gerada por PFT. Resultados experimentais mostraram que a heurística proposta, em geral, superou as heurísticas de Pan e Wang (2012). Os autores também propuseram algumas versões do IG com salto de iteração (IG_IJ), IG com busca de inserção referenciada (IG_RIS). IG_IJ e IG_RIS usam um mecanismo que escolhe, dada uma probabilidade, entre as buscas locais de inserção ou troca entre pares de tarefas. Além disso, ambas as metaheurísticas usam o critério de aceitação de SA, caso a nova solução seja pior que a atual. Resultados computacionais mostram que IG_RIS

e IG_IJ superaram todas as versões do IG de Ribas et al. (2011), HDDE de Wang et al. (2010) e algumas versões do VNS de Ribas e Companys (2013).

Toumi et al. (2017) propuseram um B&B para o problema $F_m|block|C_{max}$. Os autores desenvolveram limites inferiores baseados no Algoritmo Gilmore e Gomory (LBGG) e Ronconi (2005) (LBRQ). Os autores estabelecem um número de nós para usar os dois limitantes inferiores para o B&B proposto. Resultados computacionais mostram a superioridade do limitante inferior baseado no algoritmo de Gilmore e Gomory em comparação aos outros existentes da literatura.

Liu et al. (2018) desenvolveram uma versão do algoritmo *Biogeography-based Optimization* para o problema $F_m|block|C_{max}$. BBO é uma metaheurística baseada em população inspirada no modelo biogeográfico, que descreve como as espécies são produzidas, migradas e destruídas. Em BBO, cada indivíduo é chamado de habitante e possui um índice *fitness* de habitante (HIS), associada a função-objetivo (C_{max}). Os autores aplicaram o método NEH e aleatório para gerar a população inicial. Na próxima fase do IBBO, as soluções são ordenadas em ordem decrescente de qualidade. Na operação de migração, as piores soluções são habilitadas para obter informações adicionais das melhores taxas de imigração e emigração de soluções. A fase de mutação é composta pelo operador de troca entre pares de tarefas. Finalmente, uma vizinhança de inserção é usada para melhorar a melhor solução da população. Resultados computacionais mostram que o IBBO superou B&B de Ronconi (2005).

Panwalkar e Koulamas (2018) revisaram as formas de representação gráfica para problemas *flow shop* com bloqueio, transferência síncronas, *no-wait*, permutacional e com *buffers* limitados. Dado que na maior parte da literatura o problema $F_m|block|C_{max}$ é representado por um grafo de redes com tempos de processamento atribuídos aos seus arcos, os autores propuseram um novo grafo de redes onde o bloqueio e os tempos de processamento estão contidos no nó.

Shao et al. (2018) propuseram um *Estimation Distribution Algorithm* com VNS (EDA-VNS) para o problema $F_m|block|C_{max}$. A população inicial é gerada por meio dos métodos heurísticos NEH e aleatório. A fase de seleção usa um operador de classificação linear modificada para selecionar os melhores indivíduos. Um método probabilístico é proposto para guiar o algoritmo a espaços de solução mais promissores. O método *path relinking* é usado para aprimorar a propriedade de convergência do EDA. Para melhorar a capacidade de exploração, uma busca local referenciada modificada é aplicada aos indivíduos. Por fim, um mecanismo de manutenção da diversidade é adotado para evitar a deterioração da população. Resultados experimentais mostraram que EDA-VNS gerou melhores resultados em comparação com o

HDDE (WANG et al., 2010), hmgHS (WANG et al., 2011), IABC (HAN et al., 2012), DE-ABC (HAN et al., 2015), MFFO (HAN et al., 2016), IG (RIBAS et al., 2011), TPA (WANG et al., 2012), RAIS (LIN, S. W.; YING, 2013), SVNS (RIBAS; COMPANYS, 2013), MMA (PAN et al., 2013) e B-IG (DING et al., 2015).

Ozolins (2019) propôs um algoritmo de programação dinâmica para o problema $F_m|block|C_{max}$. Dois limitantes inferiores são propostos, baseados em Bautista et al. (2012) e Gilmore e Gomory (1964). Os resultados experimentais mostraram que, em média, o limitante inferior proposto superou o limitante inferior de Ronconi (2005).

2.3.1.2 Problema *flow shop* com bloqueio, restrições adicionais e minimização de C_{max}

Logendran e Sriskandarajah (1993) propuseram uma heurística construtiva para o problema $F_2|block, st_{ant}|C_{max}$. Os autores provaram que o problema é fortemente NP-hard. O método desenvolvido para solucionar o problema, denominado *H*, aplica o algoritmo de Gilmore e Gomory para encontrar uma sequência. Os autores estabelecem uma análise de pior caso para a heurística. O algoritmo *H* foi comparado com H_R , versão de *H* que gera seqüências aleatórias. Os resultados mostraram que, no caso médio, o algoritmo *H* superou a FC.

Levner et al. (1995) estudou uma célula robótica como um problema *flow shop* com duas máquinas, bloqueio, armazéns automatizados e estações de coleta para minimização de C_{max} . Os autores mostraram que a adição de tempos de transporte dependentes das tarefas ao problem estudado pode ser resolvido em tempo cúbico usando uma estrutura especial do algoritmo de Gilmore-Gomory.

Grabowski e Pempera (2000) propuseram uma versão de *Tabu Search* (TS) para o problema $F_m|permu, no-wait, block|C_{max}$. Os autores apresentaram duas propriedades do problema no intuito de eliminar movimentos que não melhoram o C_{max} . No TS proposto, NEH é usado para gerar a solução inicial. A vizinhança é definida de acordo com a aplicação da busca local de vizinhança de inserção, que gera $(n - 1)^2$ vizinhos. Uma redução da vizinhança é realizada aplicando as propriedades propostas. Além da aplicação de TS em um estudo de caso, resultados computacionais mostraram que TS superou SPIRIT de Widmer e Hertz (1989).

Soukhal et al. (2005) apresentaram alguns resultados de complexidade para o problema *flow shop* com duas máquinas e tempos de transporte. O problema é denotado como $F_2 \rightarrow D|block, v, c_i, t_k|C_{max}$, onde $F_2 \rightarrow D$ significa que os trabalhos são processados

primeiramente no *flow shop* de duas máquinas e após isso entregue aos clientes, onde v é o número de caminhões idênticos, c_i representa a capacidade dos caminhões e t_k o tempo o tempo de ida e volta do caminhão da fábrica ao cliente. Os autores provaram que os problemas $F_2 \rightarrow D | block, v=1, c_i=2 | C_{max}$, $F_2 \rightarrow D | block, v=1, c_i=2, t_j=h | C_{max}$ and $F_2 \rightarrow D | block, v=1, c_i=3 | C_{max}$ são fortemente NP-difíceis.

Yuan et al. (2007) estende alguns resultados de complexidade de Soukhal et al. (2005) para o problema $F_2 \rightarrow D | block, v, c_i, t_k | C_{max}$. Os autores provaram que $F_2 \rightarrow D | block, v=1, c_i=2 | C_{max}$ é NP-hard binário e $F_2 \rightarrow D | block, v=1, c_i=3, t_j=h | C_{max}$ é fortemente NP-difícil.

Liu e Kozan (2009) propuseram uma heurística construtiva, denominada algoritmo Liu-Kozan, para o problema $F_m | prmu, no-wait, block, b_{jj+1} | C_{max}$. Os autores desenvolveram alguns procedimentos para calcular a data de término de cada condição de *buffer* (*prmu*, *no-wait*, *block* and b_{jj+1}) e um procedimento para ajustar a data de início e de conclusão das condições de *buffer* anteriormente citadas. O Algoritmo de Liu-Kozan constrói a sequência de acordo com o mecanismo *Best Insertion Heuristic* (BIH), onde todas as tarefas são inseridas em todas as posições da sequência parcial. A sequência parcial com a i -ésima tarefa que minimiza C_{max} é escolhida para a próxima. A heurística termina quando todas tarefas são programadas. Resultados computacionais mostraram o desempenho do algoritmo em *flow shop* com diferentes condições de *buffers*.

Carrier et al. (2010) propuseram um *Branch & Bound* (B&B) e um GA para o problema de *scheduling* em uma célula robótica, especificado como $F_m | block, t_k | C_{max}$, onde t_k é definido como tempos de transporte da tarefa k de uma célula para outra. Os autores propuseram limitantes inferiores baseados em uma máquina e em duas máquinas para serem aplicados no B&B proposto. No GA proposto, os operadores *similar order* e troca entre pares de tarefas são aplicados nas fases de cruzamento e mutação, respectivamente. Resultados experimentais mostraram que o GA requer menos tempo computacional e é útil para resolver problemas de grandes instâncias.

Lee et al. (2010) propuseram um B&B para o problema $F_2 | block, ap_{ij} | C_{max}$. Os autores desenvolveram algumas regras de dominância para a eliminação de nós e um limitante inferior para acelerar o processo de busca do B&B. Algumas heurísticas são propostas para o problema. A primeira heurística (H1) é baseada na regra de Johnson. O segundo algoritmo, H2, visa minimizar o impacto do tempo ocioso da próxima tarefa programada. O terceiro método, H3, visa minimizar o C_{max} da sequência parcial. O quarto método, H4, visa reduzir os tempos

de bloqueio da primeira máquina ou o tempo ocioso da segunda máquina. Resultados experimentais indicaram que o B&B pode resolver problemas com até 32 tarefas em um tempo computacional razoável. Em relação aos métodos heurísticos, H4 foi o melhor avaliado.

Chen et al. (2014) propuseram uma versão híbrido do algoritmo *Differential Evolution* (DE) para o problema $F_2|block, batch, r_j|C_{max}$. A população inicial é estabelecida de acordo com a regra LPT, SPT, regra de Johnson e método aleatório. Para compor os lotes, a heurística *First-Fit* (FF) é proposta e a regra do menor tempo ocioso/de bloqueio (LIBT) é usada para programar os lotes. Uma busca local de inserção é aplicada como forma de melhora a qualidade da solução. Resultados experimentais mostraram que o proposto HDDE superou SA de Manjeshwar et al. (2009) e GA de Manjeshwar et al., (2011).

Al-Salem e Kharbeche (2016) propuseram métodos heurísticos para células robóticas, definidas como um *flow shop* com duas máquinas com tempos de processamento controlados. O problema pode ser especificado como $F_2|batch(1), block, p_{ij}^{min} \leq p_{ij} \leq p_{ij}^{max}|C_{max}$, onde *batch(1)* indica que a máquina 1 processa os trabalhos em lote e $p_{ij}^{min} \leq p_{ij} \leq p_{ij}^{max}$ é o intervalo dos tempos de processamento. Dado o MILP proposto para o problema, os autores propuseram uma heurística baseada em MIP, que consiste em resolver o problema relaxado do MILP programando conjuntamente a sequência de tarefas e o conjunto de movimentos do robô. Os autores também propuseram um GA para programar a sequência e os movimentos do robô. Resultados experimentais mostraram a eficácia e eficiência do MILP e heurísticas propostas.

Fu et al. (2017) propuseram uma Otimização de Reação Química com Busca de Vizinhança Variável (CRO-VNS) para o problema $F_m|block, ap_{ij}|C_{max}$. O algoritmo CRO é um algoritmo baseado na população que simula microscopicamente um conjunto de moléculas e tenta capturar a energia no processo de reação. Cada solução é representada por uma molécula. Quatro reações elementares são usadas para tentar otimizar o problema: colisão ineficaz *on-wall*, decomposição, colisão inter-molecular ineficaz e reações de síntese. Na proposta CRO-VNS, na fase de colisão ineficaz *on-wall*, as moléculas são modificadas por meio das buscas locais de inserção e troca entre pares de tarefas. Na decomposição, uma molécula é dividida em duas ou mais, trocando pares de tarefas aleatoriamente. Na reação de síntese, duas moléculas são cruzadas de acordo com suas distâncias, formando uma nova molécula. Finalmente, um VNS é aplicado à melhor solução encontrada. Resultados computacionais mostraram que o CRO-VNS superou a versão padrão do CRO e a versão clássica do GA.

Khorasanian e Moslehi (2017) forneceram alguns resultados para o problema $F_2|block, mtlx(1), prmp|C_{max}$. Um limite inferior para o C_{max} é proposto para diminuir o espaço

das soluções. Os autores provaram que o problema é fortemente NP-hard. Dois MILP são propostos, usando as propriedades desenvolvidas. Um VNS básico e um *Dynamic* VNS (DVNS) são propostos para o problema. A solução inicial é dada por um NEH modificado. Na fase de agitação, cinco vizinhanças baseadas em buscas locais de inserção e troca entre pares de tarefas são executadas em tarefas escolhidas aleatoriamente. Na fase de busca local, uma estrutura de vizinhança de inserção é usada. Experimentos computacionais mostraram que DVNS superou hmgHS de Wang et al. (2011), TPA de Wang et al. (2012) e HVNS de Moslehi e Khorasanian (2014) em instâncias de grande porte com 75 trabalhos ou mais.

Takano e Nagano (2017) propuseram um B&B e um (MILP) para o problema $F_m|block, s_{i,j,k}|C_{max}$. O B&B proposto segue a estrutura do algoritmo de Ronconi (2005) e quatro limites inferiores foram desenvolvidos. Ambos os métodos foram implementados e testados uns contra os outros. Resultados computacionais mostram que, em geral, o B&B proposto superou o MILP.

Koulamas e Panwalkar (2018) desenvolveram uma regra baseada em índice para o problema $F_m|no-wait, ord|C_{max}$. Os autores estenderam a aplicação da regra para o problema $F_m|block, ord|C_{max}$. Com base em resultados anteriores de desaceleração de operações proposta por Abadi et al. (2000), a regra pode obter boas soluções para o problema.

Shao et al. (2018) propuseram uma versão do algoritmo Water Wave Optimization (WWO) para o problema $F_m|block, st_{sd}|C_{max}$. Duas heurísticas são propostas pelos autores: PFT-NEH(x) e PFT-RZ(x), com base nas heurísticas PFT, NEH e RZ de Rajendran e Ziegler (1997). WWO é um algoritmo de otimização de população inspirado no modelo de ondas superficiais. O processo de busca de WWO é dado por três operadores: propagação, quebra e refração. No DWWO proposto, cada sequência é representada por uma onda (indivíduo). Para compor a população inicial, PFT-RZ(x) e um mecanismo de perturbação são usados. Na fase de propagação, um mecanismo de perturbação é usado para gerar novas ondas e uma busca local denominada Referenced Insertion Search (RIS) é aplicada a cada nova onda gerada. Na fase de refração, o método *path relinking* é proposto. Finalmente, um VNS é aplicado na fase de quebra. Resultados computacionais mostraram que o DWWO superou 23 metaheurísticas propostas para os problemas $F_m|block|C_{max}$, $F_m|st_{sd}|C_{max}$ e $1|st_{sd}|C_{max}$.

Takano e Nagano (2019) avaliaram o desempenho das principais heurísticas construtivas da literatura para resolver o problema $F_m|block, st_{sd}|C_{max}$. Os autores adaptaram métodos desenvolvidos para o problema $F_m|block|C_{max}$, proposto por McCormick et al. (1989), Ronconi (2004) e Pan e Wang (2012). Dentre os 128 métodos avaliados, resultados

computacionais mostraram que a versão adaptada de PF de McCormick et al. (1989), denominada PF1, obteve o melhor desempenho.

2.3.1.3 Problema *flow shop* com diferentes tipos de bloqueio e minimização de C_{max}

Martinez et al. (2006) apresentaram alguns resultados de complexidade para o problema *flow shop* de m máquinas com condições de bloqueio do tipo *block* e *RCb* para minimização de C_{max} . Os autores mostram que $F_2|RCb|C_{max}$, $F_3|RCb|C_{max}$, $F_4|RCb(1,2), block(2,3), Rcb(3,4)|C_{max}$, $F_3|RCb(1,2), block(2,3)|C_{max}$ e $F_3|block(1,2), Rcb(2,3)|C_{max}$ são problemas polinomiais. Os autores também mostram que $F_5|RCb|C_{max}$, $F_4|RCb(1,2), Rcb(2,3), block(3,4)|C_{max}$, $F_4|block(1,2), Rcb(2,3), Rcb(3,4)|C_{max}$ e $F_4|block(1,2), Rcb(2,3), block(3,4)|C_{max}$ são problemas NP-difíceis.

Sauer e Sauvey (2012) propuseram um GA para o problema $F_m|RCb|C_{max}$. O GA proposto usa uma associação de meta-gene, que detecta genes potenciais dos melhores indivíduos da população com o objetivo de identificar “bons” padrões de associação. GA foi comparado com NEH adaptada por Martinez (2005), heurística HSS2 por Sauvey e Sauer (2009), SA por Martinez et al. (2004) e a metaheurística *eletro-magnetism like optimization* de Yuan e Sauer (2007).

Trabelsi et al. (2011) apresentaram alguns resultados de complexidade para o problema $F_m|RCB, block, Rcb^*|C_{max}$. Os autores provaram que os problemas $F_4|RCb(1,2), block(2,3), Rcb(3,4)|C_{max}$ e $F_3|RCb^*|C_{max}$ são problemas polinomiais. Os autores desenvolveram um modelo MILP para o problema $F_m|RCB, block, Rcb^*|C_{max}$. Resultados computacionais mostram que os problemas estudados são, no máximo, menos complicados do que problemas com apenas uma restrição de bloqueio.

Trabelsi et al. (2012) propuseram heurísticas construtivas e uma versão do GA para o problema $F_m|Wb, block, Rcb^*, Rcb|C_{max}$. Os autores propuseram algumas buscas locais baseadas em TSS e NEH. TSS escolhe a próxima tarefa da sequência de acordo com a soma mínima de C_{max} parcial, a soma de tempos inativos e a soma das tarefas já programadas. Resultados computacionais mostraram que ambas as heurísticas são competitivas. Os autores também mostram que o GA proposto é eficiente para um pequeno número de tarefas.

Khorramizadeh e Riahi (2015) propuseram um ABC para o problema $F_m|Wb, block, Rcb^*, Rcb|C_{max}$. Como população inicial, os autores estabeleceram a aplicação

dos métodos NEH e aleatório. Na fase de abelha campeira, todas as p soluções são perturbadas pela inserção de uma tarefa em uma posição aleatória, gerando p' . A técnica *path relinking* é aplicada a p' para gerar um certo número de novas soluções. Uma busca local é aplicada às melhores soluções geradas. Na fase de abelha observadora, uma busca local de inserção e troca entre pares de tarefas é usada em um número determinado de iterações. Um método combinado baseado em matrizes ortogonais de Taguchi é usado na fase de abelhas. Resultados experimentais mostraram que o ABC proposto superou o GA por Trabelsi et al. (2012).

Riahi et al. (2017) propuseram uma versão do algoritmo *Scatter Search* (SSA) para o $F_m|Wb,block,RCb^*,RCb|C_{max}$. No SSA proposto, a população inicial é composta por uma solução gerada por uma versão modificada do NEH-WPT, proposta por Wang et al., (2011). O algoritmo *Iterated Local Search* (ILS) com mecanismo de inserção NEH e buscas locais de inserção e troca entre pares de tarefas são aplicadas na fase de melhoria do SSA. Na fase de combinação, a SSA aplica uma versão adaptada do *path relinking* proposto por Khorramizadeh e Riahi (2015). Resultados computacionais mostraram que SSA superou GA de Trabelsi et al. (2012) e ABC de Khorramizadeh e Riahi (2015).

Riahi et al.(2018) propuseram três versões do algoritmo *Constraint Guided Local Search* (CGLS) para o problema $F_m|st_{sd}, RCb, RSb|C_{max}$. As três versões de CGLS partem de uma solução inicial e um VNS é usado para melhorar as soluções até que um critério de parada seja alcançado. Em CGLS(1) é utilizado uma variação de NEH, denominada NNEH, proposta por Riahi et al. (2017). Em CGLS(2) e CGLS(3), a sequência inicial é gerada por uma versão modificada do NEH. O algoritmo VNS é usada em todas as versões do CGLS. Resultados experimentais mostraram que todas as buscas locais propostas superam significativamente IG de Ruiz e Stützle (2008) e GRASP de Ribas e Companys (2015).

2.3.1.4 Problema *flow shop* com *buffers* limitados e minimização de C_{max}

Dutta e Cunningham (1975) propuseram uma versão de programação dinâmica para o problema $F_2|b_{j,j+1}|C_{max}$ com $0 \leq b < \infty$. Os autores estendem alguns resultados de Reddi e Ramamoorthy (1972) para o caso $b = 0$. Para $b = 0$ ou 1, apenas soluções relativamente fracas foram obtidas, dado que o C_{max} resultou em 15% maior que o ideal.

Papadimitriou e Kanellakis (1980) apresentaram alguns resultados de complexidade para o problema *flow shop* com *buffers* limitados. Os autores provaram que o problema $F_2|b_{j,j+1}|C_{max}$ com $0 < b < \infty$ é NP completo.

Leisten (1990) propuseram duas heurísticas construtivas para o problema $F_m|b_{j,j+1}|C_{max}$ com $0 < b < \infty$. A primeira heurística (BFSE) é desenvolvida para maximizar o uso do *buffer* e evitar o bloqueio na primeira máquina. A segunda heurística, denominada BFSP, gera duas seqüências, uma do início da seqüência J e outra da seqüência final J' . Em ambas as seqüências, as tarefas são selecionadas de acordo com a diferença mínima entre o tempo de conclusão de tarefas adjacentes. Os resultados mostraram que tanto o BFSE quanto o BFSP são competitivos com NEH de Nawaz et al. (1983).

Norman (1999) propôs um *Tabu Search* (TS) para o problema $F_m|b_{j,j+1},st_{sd}|C_{max}$. O autor considerou como vizinhança todas as $(n - 1)^2$ seqüências geradas pela busca local de inserção. Resultados computacionais mostraram que TS superou NEH de NAWAZ et al. (1983), Profile Fitting (PF) de McCormick et al. (1989), BFSE de Leisten (1990) em todos os cenários de $0 \leq b_{j,j+1} \leq b$ avaliadas.

Nowicki (1999) propôs uma versão de *Tabu Search* (TS) para o problema $F_m|b_{j,j+1}|C_{max}$. O autor descreve algumas propriedades da teoria dos blocos, que são aplicadas à busca local de vizinhança de inserção para remover “movimentos inúteis”. O algoritmo proposto utiliza uma versão da heurística NEH como solução inicial, busca de vizinhança de inserção para compor a vizinhança e memórias de busca de curto e longo prazo para armazenar movimentos. Resultados computacionais mostraram que quando $b > 0$, TS superou SA e SA rápido de Van et al. (1990).

Li e Tang (2005) estenderam os estudos de Grabowski e Pempera (2000) e apresentaram novas propriedades para o problema $F_m|b_{j,j+1}|C_{max}$. Os autores também propuseram um método de aceleração para calcular C_{max} em movimentos de vizinhança de inserção. As propriedades são aplicadas na TS proposta, que utiliza NEH como solução inicial, vizinhança baseada em movimentos de inserção, com dimensão reduzida de acordo as propriedades desenvolvidas. Três versões de TS são testadas: TS^N que usa a vizinhança clássica, TS^{H1} com a nova vizinhança e TS com a vizinhança clássica e com o método de aceleração. Resultados computacionais mostram que o TS^{H1} superou as outras versões propostas de TS.

Wang et al. (2006) desenvolveram um GA híbrido (HGA) para o problema $F_m|b_{j,j+1}|C_{max}$. Os autores utilizaram as propriedades propostas por Nowicki (1999) ao algoritmo proposto para eliminar “movimentos inúteis” na vizinhança criada por uma busca

local de inserção. O HGA usa várias operações de cruzamento de forma híbrida e paralela, ou seja, um operador é escolhido aleatoriamente para executar o cruzamento. Na próxima etapa, o algoritmo escolhe entre aplicar a fase de mutação ou uma busca local, dada uma probabilidade d_p . Os autores também aplicaram diferentes operadores de mutação de inserção, inversão e troca entre pares de tarefas ao problema. Resultados computacionais mostraram que HGA supera o TS de Nowicki (1999) e Simulated Annealing (SA) de Ogbu e Smith (1990).

Liu et al. (2008) propuseram uma versão híbrida da metaheurística *Particle Swarm Optimization* (PSO) para o problema $F_m|b_{j,j+1}|C_{max}$. HPSO incorpora a busca local com aplicação das propriedades de blocos proposta por Nowicki (1999). Um *Simulated Annealing* (SA) com várias buscas locais com uma estratégia de aprendizado meta-lamarckiana adaptativa também é incorporada ao HPSO proposto. A meta-lamarckiana adaptativa determina qual busca local (inserção, troca entre pares de tarefas ou inversa) é escolhida para ser aplicada, a fim de manter a diversidade da população. Resultados experimentais mostraram que HPSO é competitiva em comparação com TS de Nowicki (1999), PSO com Variable Neighborhood Search (PSOVNS) de Tasgetiren et al. (2004) e HGA de Wang et al. (2006).

Qian et al. (2009) propuseram uma versão híbrida da metaheurística DE para o problema $F_m|b_{j,j+1}|C_{max}$. O HDE proposto usa as mesmas configurações baseadas em uma versão DE básico. Além disso, uma busca local de inserção é aplicada à melhor solução encontrada. Resultados computacionais mostraram que, no caso $b = 0$, o HDE proposto superou o RON de Ronconi (2005).

Pan et al. (2011) propuseram uma versão híbrida do algoritmo *Differential Evolution* (HDDE) para o problema $F_m|b_{j,j+1}|C_{max}$. Um procedimento para criar a população inicial é desenvolvido, que usa NEH para proporcionar uma maior diversidade e um certo nível de qualidade de indivíduos. Uma busca local é aplicada após a fase de cruzamento e é composta por um mecanismo que usa uma busca local de inserção e troca entre pares de tarefas para melhorar a sequência. Resultados computacionais mostram que o HDDE superou o Híbrido GA (HGA) de Wang et al. (2006), Híbrido PSO (HPSO) de Liu, Wang e Jin (2008), P_DDE de Pan et al. (2008) e W_DDE de Wang et al. (2010).

Pan et al. (2011) propuseram uma versão de *Harmony Search* denominada Chaotic HS (CHS) para o problema $F_m|b_{j,j+1}|C_{max}$. CHS aplica uma busca local caótica à solução selecionada. Além disso, um mecanismo baseado em *Simulated Annealing* (SA-Based) é usado para aceitar uma solução com qualidade inferior. Resultados computacionais mostraram que,

no caso $b = 0$, CHS supera o HS Melhorado de Mahdavi et al. (2007), HGA de Wang et al. (2006), mgHS de Omran (2008) e HPSO de Liu, Wang e Jin (2008).

Tang e Tang (2011) propuseram um MILP e uma versão de PSO para o problema $F_2|b_{j,j+1}, batch, st_{si}|C_{max}$. Uma versão padrão do PSO é proposta e uma busca local é incorporada ao algoritmo para tentar melhorar a posição das partículas. Resultados computacionais mostraram que ambas as abordagens de soluções propostas alcançaram soluções efetivas.

Moslehi e Khorasanian (2014) propuseram uma versão híbrida do algoritmo *Variable Neighborhood Search* (HVNS) para o problema $F_m|b_{j,j+1}|C_{max}$. O HVNS usa o NEH-WPT para gerar a solução inicial. Para gerar um novo vizinho, uma fase de agitação aplica as buscas locais de inserção ou troca entre pares de tarefa à solução inicial. Depois, uma busca local baseada em *Simulated Annealing* (SA) é usada para explorar novas vizinhanças. Resultados experimentais mostraram que, no caso $b = 0$, HVS superou HGA de Wang et al. (2006), HPSO de Liu, Wang e Jin (2008), CHS de Pan et al. (2011), hmgHS de Wang et al. (2011), TPA de Wang et al. (2012) e RAIS de Lin e Ying (2013).

Zhao et al. (2014) propuseram um PSO (LDPSO) aprimorado para o problema $F_m|b_{j,j+1}|C_{max}$. A fim de resolver ao *trade-off* entre taxa de convergência do sistema, os autores adicionaram um mecanismo de perturbação que decresce linearmente para atualizar a fórmula de velocidade. Resultados experimentais mostram que o LDPSO proposto foi superior em comparação com outros algoritmos.

2.3.2 Problema *flow shop* com bloqueio e minimização de tempo total de fluxo

Do total de artigos cobertos, 11% adotaram funções-objetivo baseadas em tempo total de fluxo ($\sum_i C_i$) como minimização da função de objetivo único. Como mostrado na Tabela 3, todos os documentos listados lidaram com o desenvolvimento de métodos de solução.

Tabela 3 - Publicações que adotaram funções baseadas no tempo de fluxo (Continua...)

Referências	Problemas	Comentários
Weng (2000)	$F_m block \sum_i C_i/n$	TS
Wang et al. (2010)	$F_m block \sum_i C_i$	HS
Deng et al. (2012)	$F_m block \sum_i C_i$	ABC

(Conclusão.)

Referências	Problemas	Comentários
Khorasanian e Moslehi (2012)	$F_m block \sum_i C_i$	IG
Han et al. (2013)	$F_m block \sum_i C_i$	ABC
Li et al. (2013)	$F_m block \sum_i C_i$	Programação dinâmica heurística
Moslehi e Khorasanian (2013)	$F_m block \sum_i C_i$	B&B
Toumi et al. (2013d)	$F_m block \sum_i C_i$	B&B
Toumi et al. (2013c)	$F_m block \sum_i C_i$	B&B
Vo et al. (2015)	$F_m permu,\beta_i \sum_i C_i$	B&B
Ribas et al. (2015)	$F_m block \sum_i C_i$	ABC
Ribas; Companys (2015)	$F_m block \sum_i C_i$	GRASP
Fernandez-Viagas et al. (2016)	$F_m block \sum_i C_i$	Heurísticas construtivas
Tasgetiren et al. (2016)	$F_m block \sum_i C_i$	VBIH
Ren et al. (2018)	$F_m block,r_i \sum_i C_i^3$	GA, TS and SA

Abreviaturas: B&B – *branch & bound*; TS – *Tabu Search*; GA – *Genetic Algorithm*; GRASP – *Greedy Randomized Adaptive Search Procedure*; HS – *Harmony Search*; IG – *Iterated Greedy*; ABC – *Artificial Bee Colony*; VNS – *Variable Search Neighborhood*; VBIH – *Variable Block Insertion Heuristic*.

Fonte: Autor, 2019.

Weng (2000) propôs um *Tabu Search* (TS) para o problema $F_m|block|\sum_i C_i/n$. Como solução inicial, o autor aplicou uma versão modificada do NEH desenvolvida para minimizar o tempo médio total de fluxo. A vizinhança considerada é a troca entre pares de tarefas. Resultados computacionais mostram que à medida que o número de *buffers* ou o número de máquinas aumenta, TS tende a diminuir seu desempenho.

Wang et al. (2010) propuseram algumas versões de *Harmony Search* (HS) para o problema $F_m|block|\sum_i C_i$. Na versão básica de HS, a memória harmônica é composta pelas soluções geradas por NEH-WPT e aleatoriamente. Na consideração da memória, uma harmonia é selecionada aleatoriamente da memória harmônica. Um ajuste do tom é aplicado à harmonia dada uma probabilidade. No algoritmo *Global HS* (gHS), funções de ajuste do tom propostas na literatura são incorporadas. Na versão *modified global best HS* (mgHS), uma nova função de ajuste de inclinação é proposta. Todas as versões híbridas de HS (hHS), gHS (hGHS) e mgHS (hmgHS) aplicam uma busca local de vizinhança de inserção para intensificar a exploração. Resultados computacionais mostram que o hmgHS obteve o melhor desempenho.

Deng et al. (2012) propuseram uma versão de *Artificial Bee Colony* (ABC), denominada *Discrete ABC* (DABC) para o problema $F_m|block|\sum_i C_i$. O NEH-WPT e o método aleatório são usados para gerar a população inicial. Na fase de abelha campeira, um procedimento baseado na melhor inserção aleatória é proposto. Uma busca local combinada com pesquisa de vizinhança de inserção e troca é aplicada na fase de abelha observadora. Na

fase da abelha exploradora, uma inserção é aplicada à melhor solução da população gerando uma nova fonte de alimento. Uma busca local da vizinhança de inserção é aplicada à nova fonte de alimento. Resultados experimentais mostraram que o DABC superou o DABC de Tasgetiren et al. (2011), HDDE de Wang et al. (2010) e *Iterated Greedy* (IG) de Ruiz e Stützle (2007).

Khorasanian e Moslehi (2012) propuseram a heurística construtiva NEH-MK e duas versões de IG para o problema $F_m|block|\sum_i C_i$. Na heurística NEH-MK proposta, as tarefas são classificadas em ordem não decrescente do tempo total de processamento. Então, o mecanismo de inserção NEH é aplicado à sequência inicial. Em cada tarefa inserida, todas as tarefas já programadas são reinseridas na melhor posição, uma por uma, com exceção da nova tarefa. Finalmente, buscas locais são aplicadas para tentar melhorar a sequência final. A solução dada por NEH-MK é usada como *input* para as duas versões propostas do IG. Resultados experimentais mostram que ambos os IGs superaram o hmgHS de Wang et al. (2010).

Han et al. (2013) desenvolveram algumas versões da metaheurística ABC para o problema $F_m|block|\sum_i C_i$. No *Discrete ABC* (DABC) proposto, a população inicial é gerada pelo método aleatório. No estágio de abelha campeira, os operadores de inserção e de troca são usados para gerar uma lista de vizinhos (NL) e a seguir uma estratégia auto-adaptativa é aplicada a NL. A metaheurística *Estimation Distribution Algorithm* (EDA) é aplicada durante a fase da abelha observadora. Três versões do DABC híbrido (hDBAC) são propostas. Os autores aplicaram MME-A e MME-B para gerar soluções iniciais para as três versões do hDABC. Em hDBAC1, hdABC2 e hABC3, uma RLS é aplicada no estágio da abelha campeira, da abelha observadora e da abelha exploradora em hDBAC1, hdABC2 e hABC3, respectivamente. Resultados computacionais mostraram que as metaheurísticas propostas superaram o HS por Wang et al. (2010).

Li et al. (2013) propuseram um modelo de matemático linear usando a teoria Max-algebra para o problema $F_m|block|\sum_i C_i$. Os autores também propõem uma heurística baseada em programação dinâmica para o problema. Experimentos computacionais mostram que a HDP proposta pode produzir soluções práticas quase otimizadas.

Moslehi e Khorasanian (2013) desenvolveram alguns métodos de solução para o problema $F_m|block|\sum_i C_i$. Dois MILP são propostos, onde o primeiro modelo é baseado nos instantes de partida e o segundo modelo é baseado nos tempos ociosos e de bloqueio. NEH-MK é usado para gerar um limite superior inicial para o B&B proposto. Três limitantes inferiores são propostos, com base na subestimativa do instante de partida de todas as tarefas não programadas. Resultados computacionais mostraram a superioridade dos limitantes inferiores propostos em comparação com o limitante inferior de Ronconi e Armentano (2001).

Toumi et al. (2013c) e Toumi et al. (2013a) desenvolveram dois B&B's para o problema $F_m|block|\sum_i C_i$. Os autores propuseram limitantes inferiores baseados em Ronconi e Armentano (2001). Ambos os autores geraram os melhores resultados encontrados para o problema das instâncias de Taillard (1993) (até 200 tarefas e 20 máquinas).

Ribas e Companys (2015) desenvolveram heurísticas construtivas e algumas versões da metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) para o problema $F_m|block|\sum_i C_i$. As heurísticas construtivas são baseadas na heurística PF. PF e HPF1 utiliza elementos que consideram o tempo total de fluxo para selecionar a próxima tarefa a ser inserida. Além do índice utilizado em HPF1, HPF2 usa um índice baseado nos dados do problema para selecionar a primeira tarefa da sequência. Em NPF, NHPF1 e NHPF2, o mecanismo de inserção NEH é aplicado às soluções geradas por PF, NHPF1 e NHPF2, respectivamente. GRASP é um processo iterativo, que consiste em duas fases: construção e busca local. Na fase de construção, uma solução viável é construída, adicionando um elemento por vez. O elemento é escolhido aleatoriamente entre os componentes de uma lista de candidatos restritos. Na fase de busca local, uma estrutura de vizinhança é utilizada para tentar melhorar a solução construída. Quatro versões do GRASP são propostas e usam HPF1, HPF2, NHPF1 e NHPF2 como soluções iniciais. Resultados experimentais mostraram que todas as heurísticas apresentaram desempenhos semelhantes, sendo HPF2 e NHPF2 as melhores heurísticas para o problema. Em relação às metaheurísticas, GRASP(HPF2) e GRASP (NHPF2) foram os mais recomendados.

Ribas et al. (2015) propuseram um DABC para o problema $F_m|block|\sum_i C_i$. A população inicial do DABC é composta por soluções dadas por HPF2 e métodos aleatórios. Na fase da abelha campeira, três operadores baseados em inserção e troca entre pares de tarefas são aplicados às fontes de alimentos. Na fase de abelha observadora, os autores usam o método *path relinking* para melhorar as fontes de alimento. Na fase da abelha exploradora, a população é atualizada se as novas soluções geradas forem melhores que as piores soluções da população. Resultados computacionais mostraram que o DABC superou o HDDE de WANG et al. (2010), DABC de Deng et al. (2012) e IG de Khorasanian e MoslehI (2012).

Vo et al. (2015) propuseram alguns limitantes inferiores gerais para o $F_m|prmu,\beta_i|\sum_i C_i$, onde $\beta_i = block, Rcb^*, Rcb, delay, no-wait, stsd$ e R_{nsd} . O autor desenvolveu limitantes inferiores para o tempo total de fluxo baseados na solução de dois subproblemas: um problema semelhante à minimização do $\sum_i C_i$ para máquina única e outro semelhante ao problema TSP. Os autores usaram a teoria MaxPlus para modelar os limitantes inferiores e os aplicaram ao B&B proposto. Uma avaliação experimental foi realizada para

resolver os problemas $F_m|permu|\sum_i C_i$ and $F_m|permu,st_{sd}|\sum_i C_i$, mas os autores sugeriram o uso dos limitantes inferiores para problemas que consideram a restrições de $block$, RCb^* , RCb .

Fernandez-Viagas et al. (2016) propôs algumas versões da heurística *Beam-Search-Based* ($BS(x)$) para o problema $F_m|block|\sum_i C_i$. A heurística adota uma estratégia baseada em busca que combina a diversificação dos algoritmos baseados em população e a eficiência da heurística construtiva. Algumas versões do $BS(x)$ superaram as principais heurísticas propostas para os problemas $F_m|block|\sum_i C_i$, $F_m|block|C_{max}$ e $F_m|prmu|C_{max}$.

Tasgetiren et al. (2016) desenvolveram tPF e tPF-NEH(x), baseado em PF-NEH(x) por Pan e Wang (2012) e uma metaheurística chamada *Variable Block Insertion Heuristic* (VBIH) para o problema $F_m|block|\sum_i C_i$. Resultados computacionais mostraram que tPF-NEH(x) superou HPF2 e NHPF2 Ribas e Companys (2015). No algoritmo VBIH, um bloco de tarefas com o tamanho BS é removido da sequência atual e, em seguida, o bloco é inserido em um número determinado de posições aleatoriamente na sequência parcial. A melhor sequência avaliada é sujeita a uma busca local variável, baseada na inserção e troca entre pares de tarefas. O tamanho do bloco mantém o mesmo valor se a solução melhora a cada iteração, caso contrário, o tamanho do bloco é incrementado em um. Um critério de aceitação baseado em SA é usado para aceitar soluções inferiores. O processo termina quando o tamanho do bloco atinge o tamanho máximo do bloco. Resultados computacionais mostraram que o VBIH superou o DABC Ribas et al. (2015).

Ren et al. (2018) propuseram versões de GA, TS e SA para o problema $F_m|block,r_i|\sum_i C_i^3$. Os autores desenvolveram algumas propriedades e regras de dominância para o problema $F_2|block,r_i|\sum_i C_i^3$, que foram aplicados ao TS proposto para reduzir o espaço de busca e acelerar a convergência do algoritmo. Resultados experimentais mostraram que o desempenho de SA aplicado a dados industriais dominou as soluções geradas por TS e GA.

2.3.3 Problema *flow shop* com bloqueio e função-objetivo baseadas em datas de entrega

Nesta seção estão incluídas todas as medidas de desempenho individuais relacionadas à data de entrega das tarefas. No total, 7% dos artigos adotam atraso total ($\sum_i T_i$) ou atraso total ponderado ($\sum_i W_i T_i$) como função mono-objetivo. A Tabela 4 lista os artigos que lidam com medidas baseadas em data de entrega.

Tabela 4 - Publicações que consideraram funções baseadas em data de entrega

Referências	Problemas	Comentários
Armentano e Ronconi (2000)	$F_m block \sum_i T_i$	TS
Ronconi e Armentano (2001)	$F_m block \sum_i T_i$	B&B
Januario et al. (2009)	$F_m block \sum_i T_i$	GA
Ronconi e Henriques (2009)	$F_m block \sum_i T_i$	GRASP
Ribas et al. (2013)	$F_m block \sum_i T_i$	ILS
Toumi et al. (2013a)	$F_m block \sum_i T_i$	B&B
Toumi et al. (2013b)	$F_m block \sum_i T_i$	B&B
Nouha e Talel (2016)	$F_m block \sum_i T_i$	PSO
Nagano et al. (2017)	$F_m block \sum_i T_i$	CSA
Shao et al. (2017)	$F_m block \sum_i T_i$	IWO

Abreviações: B&B – *branch & bound*; TS – *Tabu Search*; GA – *Genetic Algorithm*; PSO – *Particle Swarm Optimization*; *Differential Evolution*; GRASP – *Greedy Randomized Adaptive Search Procedure*; ILS – *Iterated Local Search*; CSA – *Clustering Search Algorithm*; IWO – *Invasive Weed Optimization*.

Fonte: Autor, 2019.

Armentano e Ronconi (2000) propuseram uma heurística construtiva chamada LB-NEH e algumas versões de Tabu Search (TS), para o problema $F_m|block|\sum_i T_i$. O índice LB é aplicado na primeira fase do NEH e tem como objetivo ordenar as tarefas de acordo com a diferença entre a data de vencimento e o tempo total de processamento. Na *Tabu Search* (TS) proposta, LB-NEH é usado como solução inicial. Como estratégia de geração de vizinhança, é usada uma busca local de vizinhança de inserção. Os autores compararam a versão de TS com um B&B para verificar sua eficácia.

Ronconi e Armentano (2001) propuseram um B&B para o problema $F_m|block|\sum_i T_i$. É proposto um limitante inferior para $\sum_i T_i$, com base no instante de partida das tarefas ainda não programadas. Novos limites superiores para os conjuntos de problemas das instâncias de Taillard (1993) foram encontrados pelo B&B proposto.

Ronconi e Henriques (2009) desenvolveram alguns métodos heurísticos para o problema $F_m|block|\sum_i T_i$. Na heurística proposta FPD, uma sequência é construída de acordo com uma medida de prioridade F_k , que considera o perfil de tempo de processamento da última tarefa fixa na sequência parcial, bem como as datas de entrega das tarefas candidatas. O mecanismo de inserção NEH é aplicado à sequência inicial gerada por FPD. FPDNEH é usado como solução inicial para a versão proposta do algoritmo GRASP, que utiliza três estratégias para definir o comprimento de RCL e uma busca local de vizinhança de inserção para tentar melhorar a solução atual. Uma nova versão de GRASP aplica *path relinking* para intensificar a

busca na vizinhança da solução. Os autores compararam as duas versões de GRASP com a solução inicial dada pela FPDNEH e o B&B proposto por Ronconi e Armentano (2001).

Januario et al. (2009) propuseram um Algoritmo Genético (GA) para o problema $F_m|block|\sum_i T_i$. Os autores estabeleceram uma população inicial composta por soluções adaptadas do método NEH e o método aleatório. Os autores usaram os operadores *tournament selection*, *one point crossover* e *similar job* nas fases de seleção, cruzamento e mutação, respectivamente. Uma busca local de inserção também é aplicada aos indivíduos gerados nas fases de cruzamento e mutação. Após a otimização do GA, um *path relinking* é aplicado entre duas soluções da elite. Resultados computacionais mostraram que o proposto GA superou GRASP de Ronconi e Henriques (2009).

Ribas et al. (2013) desenvolveram uma hibridização dos algoritmos *Iterated Local Search* e *Variable Neighborhood Search* (ILSVNS) para o problema $F_m|block|\sum_i T_i$. No ILSVNS proposto, a solução inicial é dada por NEH com a regra *Earliest Due Date* (EDD) e NEH com a regra de FPD como sequências iniciais. Na fase de busca local, um VNS com duas estruturas de vizinhança, troca entre pares de tarefas e inserção, é aplicado. O mecanismo de perturbação é usado d vezes por meio da aplicação de operadores de inserção e troca entre pares de tarefas. Resultados computacionais mostraram que ILSVNS apresentou melhor desempenho que IG de Ribas et al. (2011) à medida que o problema se torna mais complexo.

Toumi et al. (2013a) e Toumi et al. (2013b) propuseram alguns B&B's para o problema $F_m|block|\sum_i T_i$. Toumi et al. (2013b) também desenvolveram limitantes inferiores para o problema $F_m|block|\sum_i W_i T_i$. Ambos os autores propuseram limites inferiores, baseados em Armentano e Ronconi (2000) e os compararam com o proposto por Ronconi e Armentano (2001). Além disso, os autores atualizaram as melhores soluções dos problemas das instâncias de Ronconi e Armentano (2001).

Nouha e Talel (2016) propuseram uma versão do algoritmo PSO para o problema $F_m|block|\sum_i T_i$. Para gerar o enxame inicial, um NEH modificado foi aplicado às partículas geradas. Uma operação de cruzamento com o operador *two point crossover* e uma *Referenced Local Search* (RLS) são usados para atualizar a posição e a velocidade das partículas. Resultados experimentais mostraram que o PSO proposto é competitivo com o GRASP de Ronconi e Henriques (2009).

Nagano et al. (2017) desenvolveram uma versão da metaheurística *Evolutionary Clustering Search* (ECS) para o problema $F_m|block|\sum_i T_i$. ECS é uma metaheurística híbrida, que trabalha em conjunto com algoritmos evolutivos, no intuito de agrupar soluções para

detectar regiões promissoras no espaço de busca. O ECS proposto é composto por quatro componentes independentes: Metaheurística de busca (SM), agrupamento iterativo (AI), analisador de módulo (MA) e busca local. O SM funciona como um gerador de solução que é usado ao longo das iterações. Um GA é adotado para representar SM. O AI visa agrupar soluções semelhantes, a fim de identificar um *cluster* representativo para minimizar o problema. MA é usado para avaliar cada *cluster* em intervalos regulares, a fim de indicar o melhor *cluster* promissor. Por fim, uma busca local é aplicada no *cluster* mais promissor. Resultados computacionais mostraram que o ECS superou o ILS por Ribas et al. (2013).

Shao et al. (2017) propuseram sete versões melhoradas de FPDNEH e do algoritmo *Invasive Weed Optimization* (IWO) para o problema $F_m|block|\sum_i T_i$. As diferentes versões do FPDNEH são compostas por diferentes mecanismos de desempate, utilizados durante a fase de inserção de NEH. Resultados experimentais mostraram que as heurísticas propostas superaram o tPF-NEH(x) de Tasgetiren et al. (2016). IWO é um algoritmo de otimização evolutiva que tenta imitar o comportamento das ervas daninhas no crescimento e colonização. Uma erva daninha representa uma permutação de tarefas. A metaheurística é composta por inicialização, reprodução, dispersão espacial e exclusão competitiva. Na *Discrete Self-Adaptive IWO* proposta (SaDIWO), uma semente é gerada pela heurística FPDNEH na inicialização. Na fase de reprodução, a erva produz um número de sementes de acordo com seu valor de aptidão. Na fase de dispersão espacial, um IG é aplicado em todas as soluções e um mecanismo auto-adaptativo controla o alcance das soluções vizinhas. Na exclusão competitiva, a distância entre sementes é usada para selecionar as ervas daninhas para a próxima geração. Por fim, um VNS é aplicado às soluções geradas na fase de dispersão espacial. Resultados experimentais mostraram que oSaDIWO superou o GRASP de Ronconi e Henriques (2009), GA de Januario et al. (2009), ILS e IG de Ribas et al. (2013), GA estável de Kellegöz et al. (2010), IG_RLS de Karabulut (2016) e EDA bipartidária de Shen et al. (2015).

2.3.4 Problema *flow shop* com bloqueio e minimização do tempo de ciclo

A minimização do tempo de ciclo na programação de produção pode ser vista em publicações que adotam sistemas de produção repetitivos, como linhas de fluxo de montagem. A proposição de métodos de solução e análise de complexidade de problemas para este objetivo de desempenho somam 3% do total de artigos cobertos e pode ser visto na Tabela 5.

Tabela 5 - Publicações que consideraram funções baseadas em tempo de ciclo

Referências	Problemas	Comentários
Mccormick et al. (1987)	$F_m block C_t$	O problema foi provado ser NP-completo
Mccormick et al. (1989)	$F_m block C_t$	Heurística construtiva
Kamoun e Sriskandarajah (1993)	$F_m block C_t$	$F_3 no-wait(1,2), block(2,3) C_t$ and $F_3 block C_t$ são fortemente NP-difíceis
Abadi et al. (2000)	$F_m block C_t$	Heurística construtiva
Abadi (2007)	$F_m block C_t$	Heurística construtiva

Fonte: Autor, 2019.

Mccormick et al. (1987) forneceu alguns resultados de complexidade para uma linha de produção repetitiva do tipo *flow shop* de m máquinas com minimização do tempo de ciclo. Os autores provaram que o problema $F_m|block|C_t$ é NP-Completo. Mccormick et al. (1989) estende os estudos dos autores anteriores. Os autores propuseram a heurística *Profile Fitting* (PF) para o problema $F_m|block|C_t$. Em suma, o PF seleciona a próxima tarefa da sequência de acordo com o tempo ocioso mínimo entre o último trabalho programado na sequência parcial e a tarefa atual a ser inserida.

Kamoun e Sriskandarajah (1993) estudaram a complexidade da programação de tarefas em sistemas de produção repetitivos. Os autores provaram que os problemas $F_3|no-wait(1,2), block(2,3)|C_t$ e $F_3|block|C_t$ são fortemente NP-hard.

Abadi et al. (2000) propuseram as heurísticas de *sequential slowing down* (SSD) e de *parallel slowing down* (PSD) para o problema $F_m|block|C_t$. SSD é baseada em GENIUS de Gendreau et al. (1992). Os autores provaram que para qualquer sequência δ , $C_tB(\delta)$, calculado de acordo com os aspectos de bloqueio B , é igual a $C_tNWS(\delta)$, onde NWS corresponde ao problema *no-wait flow shop*, se o tempo de processamento de algumas operações for retardado. PSD e SSD foram testados com um limite inferior e duas abordagens de decomposição.

2.3.5 Problema *flow shop* com bloqueio e função multi-objetivo

Como mostrado na Tabela 6, 12% das publicações trataram da minimização de funções multi-objetivo. A maioria propôs métodos de solução, que incluem MILP e metaheurísticas. Uma vez que algumas funções-objetivo frequentemente entram em conflito consigo mesmas, um conjunto de soluções ótimas dado pela curva de Pareto é fornecido. Assim,

um problema de otimização multi-objetivo consiste em identificar todos os elementos que pertencem ao conjunto de Pareto, que contêm todas as soluções que não são dominadas por qualquer outra solução.

Tabela 6 - Publicações encontradas que adotaram função multi-objetivo

Referências	Problemas	Comentários
Prasad et al. (2006)	$F_M B_{j,j+1} \sum_I \bar{C}_{Icont} + \sum_I \bar{C}_{Iparts} + \sum_I \sigma_I$	GA
Qian et al. (2009)	$F_M B_{j,j+1} C_{Max}$ And T_{Max} $F_M B_{j,j+1} C_{Max}$ And $\sum_I I_J$	DE
Cohn et al. (2010)	$F_M Block C_{Max}$ And CR	<i>Trade-offs</i> entre C_{max} and CR podem mudar as características da solução.
Ventura e Yoon (2013)	$F_m B_{j,j+1},lot-str \sum_I(t_i + E_i)$	GA
Nouri e Ladhari (2015)	$F_m Block C_{max}, F_m Block \sum_I c_i$ and $f_m Block \sum_I t_i$	GA e ABC
Guanlong et al. (2016)	$F_M Block C_{Max} + \sum_I C_I$	MDGSO
Nouri e Ladhari (2017)	$F_m Block C_{max} + \sum_I c_i$	GA
Lebbar et al. (2018)	$F_M Block,R_I Ac_{Max} + (\alpha-1)T_{Max}$	MILP
Gong (Han e Sun (2018)	$F_m Lot-str,block C_{max} + E_{max}$	ABC
Lebbar et al. (2018)	$F_M Block C_{Max}$ And T_{Max}	GA
Shao, Pi e Shao (2018a)	$F_M Block,R_I C_{Max} + \sum_I T_I$	IWO
Wang et al. (2018)	$F_M Block C_{Max} + E_W$	VNS
Yang e Liu (2018)	$F_m Block \bar{C}_{Max} + \tilde{A}i$	GWO
Han et al. (2019)	$F_m Brkdwn,block,lot-str C_{Max}$ And T_{Max}	REMO

Abreviações: GA – *Genetic Algorithm*; ABC – *Artificial Bee Colony*; MDGSO – *Multi-objective discrete group search optimizer*; IWO – *Invasive Weed Optimization*; VNS – *Variable Neighborhood Search*; GWO – *Grey Wolf Optimization*; REMO – *Evolutionary Robust Scheduling Algorithm*.

Fonte: Autor, 2019.

Prasad et al. (2006) consideraram o problema *flow shop* com estoques *kanban* como *buffers*. O bloqueio duplo ocorre pelo tipo de peça e tamanho da fila na máquina. Os objetivos do problema são a minimização do tempo médio de conclusão dos contentores ($\sum_i \bar{C}_{icont}$), o tempo médio de conclusão dos tipos de peças ($\sum_i \bar{C}_{iparts}$) e o desvio padrão do tempo médio de conclusão dos tipos de peças ($\sum_i \sigma_i$). Os autores propuseram uma versão modificada do Algoritmo Genético (GA), denominada “Algoritmo Genético Multiobjetivo de Ordenação por Distância Não Dominada e Normalizada” (NDSMGA). A população inicial de NDSMGA inclui a geração de indivíduos diferentes para cada função objetiva. A classificação da população é baseada no *ranking* de Pareto. O algoritmo escolhe a aplicação de cruzamento ou

mutação dada uma probabilidade. Uma fase adicional, que emprega a troca entre pares de tarefas, é aplicada a todos os indivíduos. Resultados computacionais mostraram que o NDSMGA superou os outros métodos avaliados na maioria dos problemas.

Gong e Tang (2008) propuseram uma heurística gulosa para o problema $F_m|block, t_k, batch(1)|C_{max} + \sum_i B_i$, onde t_k representa os tempos de transportes e $batch(1)$ define que as tarefas são processadas em lote na máquina 1. Os autores provaram que o problema considerado é fortemente NP-hard, mesmo quando o tempo de transporte é igual a zero. A heurística gulosa H usa a regra LPT para criar os lotes. Uma análise pior caso da Heurística H é teoricamente apresentada.

Qian et al. (2009) propuseram uma versão híbrida da metaheurística *Differential Evolution* (DE) para os problemas $F_m|b_{j,j+1}|C_{max}$ and T_{max} and $F_m|b_{j,j+1}|C_{max}$ and $\sum_j I_j$. Em HDE, a população inicial é gerada aleatoriamente. Além da estrutura básica do DE, uma busca local de inserção é aplicada aos indivíduos. Resultados computacionais mostraram que o HDE superou o IMMOGLS2 de Ishibuchi et al. (2003) no caso $b = 0$ para ambos $F_m|b_{j,j+1}|C_{max}$ e T_{max} and $F_m|b_{j,j+1}|C_{max}$ and $\sum_j I_j$.

Cohn et al. (2010) propuseram um estudo da relação entre C_{max} e a responsividade do cliente (CR) em um *flow shop* de m máquinas, denotado como $F_m|block|C_{max}$ e CR . Os resultados computacionais mostraram como diferentes abordagens utilizadas para atender o *trade-off* entre essas duas métricas podem alterar substancialmente as características da solução.

Gong et al. (2010) discutiram alguns resultados de complexidade e propuseram alguns algoritmos de aproximação para o problema $F_2|\beta \rightarrow \delta, block, sharedst|\lambda C_{max} + (1-\lambda)\sum_j B_j$. Os autores provaram que o problema $F_2|\beta \rightarrow \delta, block, sharedst|\lambda C_{max} + (1-\lambda)\sum_j B_j$ é fortemente NP-difícil. Dois algoritmos de aproximação e um modelo matemático são propostos para o problema de tarefas de sequenciamento e formação de lotes. Os autores concluem que os dois algoritmos de aproximação são úteis para gerar soluções de boa qualidade.

Maleki-Daroukolaei et al. (2012) propuseram um MILP para o problema $3AF|block, s_{i,k,j}|\alpha(\sum_i W_i C_i / W) + (\alpha-1)C_{max}$. Os autores também propuseram uma versão de *Simulated Annealing* (SA). Os autores concluíram que, devido a um grande número de variáveis e restrições, o SA tem um melhor desempenho computacional do que o MILP. Além disso, o SA foi capaz de resolver vários problemas em um tempo menor que o MILP.

Ventura e Yoon (2013) propuseram uma versão de GA para o problema $F_m|b_{j,j+1}, lot-str|\sum_i (T_i + E_i)$ com sublotes iguais. Na proposta NGA (NGA), a população inicial é gerada aleatoriamente. O operador *Roulette Section Wheel* é usado para selecionar os

indivíduos pais. Uma versão do operador PMX é usado na fase de cruzamento. Na fase de mutação, um movimento de troca entre pares de tarefas é realizado no indivíduo gerado na fase de cruzamento, dado uma taxa de mutação fixa. Resultados computacionais mostraram que o NGA melhorou significativamente a versão básica do GA em todas as instâncias avaliadas.

Nouri e Ladhari (2015) propuseram GA e ABC para os problemas $F_m|block|C_{max}$, $F_m|block|\sum_i C_i$ e $F_m|block|\sum_i T_i$. No GA proposto, denominado *Block GA* (BGA), a população inicial é composta por soluções geradas pelos métodos NEH-WPT, mod-NEH-MK e aleatório. Os operadores *Roulette Section Wheel* e *two point crossover* são usados nas fases de seleção e crossover, respectivamente. No *Block ABC* (BABC), a população inicial de fonte de alimentos é gerada pelo método PF-NEH(x) e aleatório. Na fase da abelha campeira, a técnica *path relinking* é aplicada para produzir novas fontes de alimento. Na fase de abelha observadora, uma RLS é aplicada a uma sequência selecionada de acordo com o operador *Roulette Section Wheel*. Na fase das abelhas exploradoras, as piores soluções são substituídas por novas fontes de alimento, aplicando a busca local de inserção na melhor solução da população. Resultados computacionais mostraram que BGA e BABC superaram MMA de Pan et al. (2013), DPSO de Wang e Tang (2012), DABC de Deng et al. (2012), IG de Ribas et al. (2011), HDDE de Pan et al. (2011), RAIS de Lin e Ying (2013), hmgHS por Wang et al. (2011), hDABC de Han et al. (2013), IG por Khorasanian e Moslehi (2012) e GRASP de Ronconi e Henriques (2009).

Guanlong et al. (2016) propuseram uma versão da metaheurística *Multi-Objective Discrete Group Search Optimization* (MDGSO) para o problema $F_m|block|C_{max} + \sum_i C_i$. GSO é uma metaheurística estocástica populacional que consiste em três funções: *Producer*, *Scrounger* e *Ranger*. Na fase de *producer* a região vizinha de uma solução relativamente melhor é explorada. Na fase de *scrounger*, as soluções são geradas por operadores de cruzamento. Na fase de *ranger*, uma busca aleatória é feita para tentar melhorar as soluções. Em MDGSO, a população inicial é constituída por indivíduos gerados por NEH, NEH-WPT e método aleatório. Neste caso, as melhores soluções são definidas como não dominadas. Na fase de *producers*, a busca local de pareto baseada em inserção é aplicada ao conjunto de soluções não dominadas (NS). Se um número aleatório for menor que uma dada probabilidade, a fase de *scrounger* será aplicada. Neste procedimento, o operador de cruzamento *partial mapped crossover* é executado para uma solução selecionada aleatoriamente do conjunto NS. Caso contrário, o procedimento de *ranger* é executado. Então, uma solução aleatória é selecionada do conjunto de NS e a busca local de vizinhança de inserção é aplicada até que nenhuma solução melhor seja encontrada. Os resultados computacionais mostraram que o MDGSO superou NDGA-II de Deb et al. (2002) e SA de Lin e Ying (2013).

Nouri e Ladhari (2017) propuseram um GA para o problema $F_m|block|C_{max} + \sum_i C_i$. Para obter a população inicial, os autores propuseram o uso de NEH-WPT, mod-NEH-MK e método aleatório. A técnica de classificação para soluções não dominadas é usada para classificar os membros da população. Na fase de seleção, o operador *binary tournament* é usado. Para gerar novos indivíduos, o operador *two-point crossover* é aplicado na fase de cruzamento. O elitismo é usado para atualizar a próxima geração. Resultados experimentais mostraram que o GA proposto superou SPEA-II de Zitzler et al. (2001).

Gong et al. (2018) propuseram uma versão do algoritmo ABC para o problema $F_m|lot-str,block|C_{max} + E_{max}$. No proposto *multi-objective* ABC (MOABC), a população inicial é obtida pela aplicação de variantes de NEH e MME e método aleatório. Na fase da abelha campeira, MOABC usa operadores de cruzamento ou mutação para gerar novas fontes com uma certa probabilidade. Um operador de construção/destruição é aplicado no estágio da abelha observadora. Para melhorar os indivíduos no estágio da abelha observadora, uma busca local de pareto é aplicada aos p_i indivíduos gerados na fase de abelha campeira. Para atualizar a população, o elitismo é usado para selecionar as melhores fontes de alimento. Os resultados computacionais mostraram que o MOABC proposto obteve um desempenho melhor do que outros algoritmos avaliados da literatura.

Lebbar et al. (2018) propuseram um MILP para o problema $F_m|block,r_i|\alpha C_{max} + (\alpha-1)T_{max}$. Os resultados experimentais mostraram que, em geral, o modelo proposto é computacionalmente eficiente para resolver o problema considerado.

Lebbar et al. (2018) propuseram uma versão melhorada do Algoritmo Genético de Ordenação Não Dominada II (NDGA-II) para o problema $F_m|block|C_{max}$ and T_{max} . NDGA-II usa o método NEH-WPT e aleatório para gerar a população inicial. Os operadores *binary tournament*, *two point crossover* e inserção são aplicados nas fases de seleção, cruzamento e mutação, respectivamente. O elitismo é usado para atualizar a população. Resultados experimentais mostraram que o NDGA-II proposto pode fornecer soluções não dominadas mais competitivas do que o HDDE Qian et al. (2009).

Shao et al. (2018b) propuseram um IWO para o problema $F_m|block,r_i|C_{max} + \sum_i T_i$. Na versão proposta, denominada *multi-objective discrete* IWO (MODIWO), a população inicial é composta por plantas daninhas geradas por PFT e FPD. Na fase de reprodução, uma RLS é usada para gerar novas sementes. Na fase espacial de dispersão, d inserções são realizadas na semente atual, onde d é um parâmetro a ser ajustado. Na fase de exclusão competitiva, a rápida estratégia de triagem não dominada e a técnica *crowding distance*, proposta por Deb et al. (2002), são aplicadas para determinar a próxima geração. Na fase de auto-adaptação, uma busca

local é aplicada nas novas sementes para melhorar sua adaptabilidade. Avaliações experimentais mostraram que o MODIWO proposto superou os mais recentes algoritmos desenvolvidos para problemas de *scheduling* multi-objetivo.

Wang et al. (2018) propuseram uma versão do algoritmo *Variable Neighborhood Search* (VNS) para o problema $F_m|block|C_{max}^+ E_W$. Uma versão de NEH é proposta para gerar a solução inicial do proposto *Multi-objective VNS* (MOVNS). As buscas locais de inserção e troca de vizinhança fazem parte da estrutura do VNS. Além disso, uma busca local de inserção é proposta para compor o algoritmo. Resultados computacionais mostraram que o MOVNS superou o NSGA-II de Deb et al. (2002) e o BMSA de Lin e Ying (2013).

Yang e Liu (2018) consideraram o problema *flow shop* de m máquinas com bloqueio e tempos de processamento e datas de entrega *fuzzy* para minimização de C_{max}^{fuzzy} e o índice médio de concordância *fuzzy*. O problema pode ser denotado como $F_m|block|\tilde{C}_{max} + \tilde{AI}$, onde AI indica o grau de conformidade entre $\tilde{C}_i$ e $\tilde{d}_i$. Uma versão da metaheurística *Grey Wolf Optimizer* (GWO) é proposta. No HGWO proposto, cada permutação é definida por um vetor de dimensão n que representa a posição de um lobo cinza. Uma população inicial é composta por soluções geradas por uma versão modificada do PFE e método aleatório. Três líderes são selecionados de acordo com o operador *Roulette Section Wheel* para orientar a população restante. Duas buscas locais de inserção são aplicadas para melhorar cada lobo. Resultados computacionais mostraram que o HGWO proposto superou o NSGAII por Deb et al. (2002).

Han et al. (2019) propuseram um algoritmo evolutivo para o problema $F_m|brkdown,block,lot-str|C_{max}$ and T_{max} . No algoritmo evolutivo proposto, denominado *evolutionary robust scheduling algorithm* (REMO), é utilizado uma variante do método NEH para gerar um indivíduo da população e o restante é gerado aleatoriamente. São propostos dois operadores de cruzamento melhorados denominados *similar block order crossover* e *similar order job crossover*. A mutação consiste em movimento de inserção ou troca entre pares de tarefas selecionadas aleatoriamente. Resultados experimentais mostraram que o REMO proposto superou as principais metaheurísticas da literatura.

2.3.6 Problema *flow shop* com bloqueio e minimização de função-objetivo estocástica

Em problemas de *scheduling* que admitem tempos estocásticos, presume-se que os tempos de processamento obedecem a uma distribuição de probabilidade conhecida e que

podem ser convertidos em determinísticos. A Tabela 7 resume as publicações que adotaram medidas estocásticas, que corresponde a 6% dos artigos revisados.

Tabela 7 - Publicações que adotaram funções-objetivo estocásticas

Referências	Problemas	Comentários
Pinedo (1982)	$F_m block E(C_{max})$	Regra de sequenciamento
Foley e Suresh (1984)	$F_2 block E(\sum_i C_i)$	Regra de sequenciamento
Wie e Pinedo (1986)	$F_m block E(C_{max})$ and $F_m block E(\sum_i C_i)$	Regra de sequenciamento
Jia (1998)	$F_2 block E(\sum_{i=1}^n \sum_{j=1}^n w_{[i]}^2 - w_{[j]}^2)$	Regra de sequenciamento
Kalczynski e kamburowski (2005)	$F_2 block E(C_{max})$	O problema é equivalente ao TSP para uma matrix Monge permutada.
Hu et al. (2008)	$F_m b_{j,j+1} E(C_{max})$	DE
Han et al. (2016)	$F_m block,lot-str E(C_{max}) + E[var(T_{max})]$	MOEA
Han et al. (2019)	$F_m block,lot-str E(\bar{C}_{max}) + E[var(C_{max})]$	MBO

Abreviações: DE - *Differential Evolution*; MOEA - *Multi-objective Evolutionary Algorithm*; MBO - *Migrating Birds Optimization*.

Fonte: Autor, 2019.

Pinedo (1982) propôs uma regra de sequenciamento para o problema $F_m|block|E(C_{max})$. Com base nos teoremas desenvolvidos, os autores recomendaram a programação de tarefas com menores tempos de processamento esperados e maiores variações nos tempos de processamento no início e no final da sequência, respectivamente, e tarefas com tempos de processamento mais longos e menores variações nos tempos de processamento no meio da sequência.

Foley e Suresh (1984) provam que a regra do Menor Tempo Total de Processamento Esperado minimiza o $F_2|block|E(\sum_i C_i)$.

Wie e Pinedo (1986) mostram alguns resultados de complexidade para os problemas $F_m|block|E(C_{max})$ e $F_m|block|E(\sum_i C_i)$. Os autores consideraram que o tempo de processamento da i -ésima tarefa em cada uma das m máquinas é igual à variável aleatória X_j e é distribuído de acordo com F_j . Assim, o tempo de processamento é estocasticamente ordenado, tal que, $F_1 \leq \dots \leq F_n$. Os autores provaram que a sequência $J_1, J_3, J_5, \dots, J_{n-1}, J_n, J_{n-2}, \dots, J_6, J_4, J_2$ quando n é par e a sequência $J_1, J_3, J_5, \dots, J_{n-2}, J_n, J_{n-1}, \dots, J_6, J_4, J_2$ quando n é ímpar minimiza $E(C_{max})$. Os autores também provaram que a sequência $J_1, \dots, J_n$ minimiza o $E(\sum_i C_i)$.

Jia (1998) apresentou alguns resultados para o problema $F_2|block|E(\sum_{i=1}^n \sum_{j=1}^n |w_{[i]}^2 - w_{[j]}^2|)$. O autor teve como objetivo determinar a sequência ótima que minimiza as diferenças

absolutas totais esperadas nos tempos de espera até que o processamento seja iniciado na segunda máquina. Os autores provaram que quando n é ímpar, a sequência ótima que minimiza a função objetivo é dada por $J_n, J_{n-2}, J_3, \dots, J_1, J_2, \dots, J_{n-3}, J_{n-1}$ e seu reverso. Quando n é par, sequência $J_n, J_{n-2}, \dots, J_2, J_1, J_3, \dots, J_{n-3}, J_{n-1}$ minimiza a função objetivo.

Kalczynski e Kamburowski (2005) forneceram alguns resultados de complexidade do problema $F_2|block|E(C_{max})$. Os autores mostraram que o problema é equivalente ao TSP à matriz Monge permutada. Os autores também identificaram uma nova classe de casos eficientemente solucionáveis.

Hu et al. (2008) propuseram uma versão do algoritmo Differential Evolution (DE) com a técnica *optimal computing budget allocation* (OCBA) e teste de hipótese (OHTDE) para o problema $F_m|b_{j,j+1}|E(C_{max})$. Em DE, uma estrutura especial de cruzamento, baseada em movimentos de troca entre pares de tarefas, é incorporada para aumentar sua capacidade de exploração. OCBA é aplicada para prever amostragens de modo a identificar boas soluções. O teste de hipótese é usado para estimar o desempenho da solução e reduzir buscas repetitivas. Resultados computacionais mostraram a eficácia dos mecanismos adotados.

Han et al. (2016) propuseram um algoritmo de otimização evolutiva multi-objetivo (MOEA) para o problema $F_m|block, lot-str|E(C_{max}) + E[var(T_{max})]$, onde var corresponde à variância. O problema é formulado como um problema de otimização multiobjetivo, onde cada objetivo de intervalo é dinamicamente convertido em um valor real usando uma soma ponderada do seu ponto médio e raio. MOEA é baseado no mecanismo de seleção do NSGA-II de Deb et al. (2002). A população inicial é gerada por uma versão modificada do MME e o restante aleatoriamente. Os autores propuseram um operador de cruzamento que considera informações de soluções não dominadas entre os pais selecionados. Durante a fase da mutação, uma busca local de inserção é proposta. Cada mutante gerado pela busca local é avaliado pela distância em relação aos seus pais. Resultados computacionais mostraram que o MOEA superou a versão melhorada de NSGA-II de Deb et al. (2002) e EDA baseada em Pareto (PBEDA) de Tiwari et al. (2015)

Han et al. (2019) propuseram uma versão do algoritmo *Migration Birds Optimization* (MBO) para o problema $F_m|block, lot-str|E(\bar{C}_{max}) + E[var(C_{max})]$, onde $\bar{C}_{max}$ é a média C_{max} das n tarefas e var corresponde à variância. MBO é uma metaheurística baseada em enxames que tenta imitar a formação de pássaros em V e consiste em quatro partes principais: a) Inicialização, que visa gerar a população inicial; b) melhoria da solução líder; c) melhoria das soluções restantes e; d) substituição da solução líder. Na MBO proposta (MOBMO), as soluções

são geradas usando uma versão modificada do PFE e aleatoriamente. Troca entre pares de tarefas, inserção ou inversão é selecionada aleatoriamente e aplicada à sequência principal. Um modelo probabilístico é desenvolvido e por meio de sua aplicação, as soluções são geradas. Além disso, uma RLS é proposta para uma solução selecionada aleatoriamente da população. Resultados computacionais mostraram que MOBMO superou NSGA-II de Deb et al. (2002), a versão básica de MOB, algoritmo de *robust evolutionary multi-objective scheduling* de Han et al. (2016), *decomposition based multi-objective evolutive algorithm* de Shen et al. (2017) e *hibrid evolutive multi-objective algorithm* de Fu et al. (2018).

2.3 CONSIDERAÇÕES FINAIS DA SEÇÃO 2

Neste capítulo, foi apresentada uma revisão de literatura envolvendo o problema *flow shop* com bloqueio. No total, a primeira revisão retornou 139 artigos, datando-se de 1969 até Maio de 2019.

Em relação as características do ambiente de máquinas, a maior parte das publicações são voltadas para o problema de n tarefas, m máquinas e somente a restrição de bloqueio. Recentemente, diferentes abordagens de bloqueio, assim como a combinação de diferentes condições de estoque intermediário em um mesmo *flow shop* estão sendo abordadas (MIYATA; NAGANO, 2019).

Em relação à função objetivo, é predominante e notório que a adoção da duração total da programação (C_{max}), por ser um dos critérios de desempenho mais estudados na área de *scheduling* determinístico. Entretanto, pode-se notar um crescimento, em especial nos últimos anos, de publicações que tratam do tempo total de fluxo, do atraso total e de funções multi-objetivo.

Em relação aos métodos de solução, dada a complexidade do problema, pode-se observar uma predominância maior em proposição de métodos heurísticos. Neste sentido, identifica-se uma tendência maior, em especial nos últimos anos, da proposição de metaheurísticas, em especial algoritmos que utilizam populações como *Genetic Algorithm*, *Harmony Search*, *Artificial Bee Colony*, *Differential Evolution* e *Particle Swarm Optimization*. Em questão de desempenho dos métodos, pode-se observar que o banco de instância de problemas de Taillard (1993) tem sido utilizado como forma de avaliação. Neste sentido, a

literatura contempla uma atualização das melhores soluções para o problema, que podem ser utilizadas também para avaliação da qualidade de solução de métodos propostos.

As seguintes oportunidades de pesquisa futuras podem ser sugeridas:

- Adição de restrições adicionais ao problema *flow shop* com bloqueio, tais como tempos de *setup*, restrições de disponibilidade de máquina, recursos de máquinas compartilhados (*single server*) e também uma forma mais ampla de avaliação do sistema de sequenciamento, como o estudo do *flow shop* distribuído;
- Estudos de cenários envolvendo estágios com e sem possibilidade de estoques intermediários entre máquinas, adicionando-se restrições adicionais;
- Proposição de métodos heurísticos para os cenários RCb e RCb*, uma vez que pode ser observado uma lacuna de métodos nesta área;
- Utilização de outras medidas de desempenho, como por exemplo, funções-objetivos sujeitas a outros critérios de desempenho como pode ser visto em Aydilek e Allahverdi (2012), Allahverdi e Aydilek (2013) e Allahverdi e Aydilek (2014);
- Proposição de métodos heurísticos para os problemas com restrições adicionais levantadas anteriormente.

3 DEFINIÇÃO DO PROBLEMA

Esta Seção apresenta a definição do problema de *scheduling flow shop* permutacional com m máquinas, com ocorrência de bloqueio, com tempos de *setup* dependentes da sequência e com inclusão de manutenção preventiva (MP) nas máquinas. O objetivo do problema é a minimização da duração total da programação e do tempo total de fluxo dado que os custos de MP não podem ser maiores que um determinado valor.

Esta Seção é estruturada em 6 subseções: Na primeira e na segunda seção são definidas as hipóteses do problema de *scheduling* e da abordagem de MP. Na terceira seção é apresentado o modelo de programação linear inteira mista (MILP) para os problemas estudados. Na quarta seção é apresentado um procedimento de cálculo das atividades de MP dada uma sequência e, adicionalmente, uma busca local para reposicionamento destas atividades é proposta. Na quinta seção é apresentado os intervalos entre MP's da literatura utilizados como parte da instância dos problemas. Por fim, na sexta seção são descritas as instâncias de problemas geradas para os problemas e formas de avaliação de desempenho dos métodos de solução propostos nas Seções 4 e 5.

3.1 PROBLEMA DE *SCHEDULING FLOW SHOP* COM BLOQUEIO E *SETUP* DEPENDENTE DA SEQUÊNCIA

O problema abordado pode ser formalmente definido como segue: um conjunto de n tarefas deve ser processado em um ambiente *flow shop* com m máquinas em série. As tarefas passam pela mesma ordem de processamento, ou seja, possuem fluxo padrão. O objetivo geral do problema é encontrar uma sequência de n tarefas que minimiza o critério de desempenho adotado dado que o custo total de manutenção preventivo da programação não seja maior que um determinado valor. Como forma de melhor definir os limites do problema, as seguintes hipóteses são adicionadas:

- A prioridade de processamento das tarefas em cada máquina segue a regra *First In First Out* (FIFO), ou seja, a sequência de tarefas estabelecida na primeira máquina é a mesma para todas as outras máquinas;

- O ambiente *flow shop* é composto por m estágios de processamento ($m \geq 2$), sendo que cada estágio é constituído por apenas uma máquina;
- Uma máquina pode processar apenas uma tarefa por vez;
- Uma tarefa pode ser processada por apenas uma máquina por vez;
- Não há estoques intermediários entre as máquinas. Neste caso, trata-se de um *flow shop* com bloqueio do tipo *block*. Então, se o processamento da tarefa i na máquina $j + 1$ não foi finalizado, a tarefa $i + 1$ permanecerá na máquina j bloqueando que uma nova tarefa seja processada até que a tarefa i inicie o processamento na máquina $j + 2$ libere a máquina $j + 1$ para processar a tarefa $i + 1$;
- Os tempos de *setup* são considerados separados dos tempos de processamento, podem ser antecipados e são dependentes da sequência. O tempo de *setup* da máquina j para uma tarefa k que é precedida pela tarefa i é denotado como $s_{j,i,k}$;
- As m máquinas estão sujeitas a manutenções preventivas (MP's);
- Duas medidas de desempenho, ou seja, dois problemas são adotados: duração total da programação (C_{max}) e tempo total de fluxo ($\sum C_i$);
- A função-objetivo está sujeita a um dado custo total de MP, ou seja, a função-objetivo é classificada como hierárquica.
- Não é permitida a preempção de tarefas, isto é, o processamento da tarefa não pode ser interrompido para processar outras tarefas ou para operação de alguma MP.

3.2 MANUTENÇÃO PREVENTIVA BASEADA EM NÍVEIS

Nesta seção é descrita a abordagem de Manutenção Preventiva (MP) para o problema estudado. A periodicidade da MP e a taxa de deterioração dos componentes das máquinas serão adotados como aspectos da inclusão da MP ao problema. É considerado que os níveis de MP (NM) vão decrescendo linearmente durante o processamento das tarefas. Os níveis máximos de MP (periodicidade da MP) são estabelecidos de acordo com dois cenários de MP apresentados na Seção 3.5, que determinam intervalos entre MP's: O Cenário I, baseado em Yu e Seif (2016), considera o tempo de deterioração máxima possível para uma dada instância e o Cenário II, proposto por Ruiz *et al.* (2007), tem como objetivo manter uma confiabilidade limiar mínima para um dado período de produção t .

As atividades de MP segue o que foi proposto por Yu e Seif (2016), ou seja, a MP é baseada na condição da máquina. Os autores definem dois conceitos para a inclusão das MP's ao problema flow shop permutacional: diversidade e flexibilidade. A diversidade significa que diferentes tipos de atividades de MP (MA's) são realizadas em uma mesma máquina, sendo que cada atividade apresenta diferentes periodicidades, dependentes da deterioração que as tarefas causam neste determinado tipo. Neste sentido, existem diferentes níveis de MP (NM's). A flexibilidade implica que as MA's podem ser realizadas a qualquer momento dentro do intervalo entre MP's de um determinado NM, não necessariamente quando a condição da máquina atinge seu nível mínimo de deterioração. Denota-se o NM t da máquina j como $NM_{j,t}$, que é representado por um número inteiro entre 0 e $NM_{j,t}^{max}$, com $NM_{j,t}^{max} > 0$. Quanto maior $NM_{j,t}$, melhor o estado de operação da máquina. Se $NM_{j,t}$ atinge $NM_{j,t}^{max}$, a máquina está em seu melhor estado. Devido à condição de não preempção, as atividades de MP (MA) do nível t na máquina j ($MA_{t,j}$) são definidas como todas as ações responsáveis por reparar a máquina e trazê-la a seu estado $NM_{j,t}^{max}$. As $MA_{t,j}$'s podem ser programadas entre as tarefas que ocupam as posições adjacentes $i - 1$ e i ($i = 1, \dots, n$) e podem ser denotadas como $MA_{[i-1],[i],j}^t$. Cada MA possui um custo associado a troca de peças, denotado como $SP_{t,j}$ e um custo de mão-de-obra (WF_t) diretamente relacionado com a duração da MA, ou seja, o custo total de mão-de-obra é definido como $WF_t \times MA_{[i-1],[i],j}^t$. O custo total de MP, $total_cost$, pode ser definido como a soma de todos os custos da programação de MP's da sequência. Cada NM é afetado quando: a) as tarefas são processadas, correspondendo a deterioração da máquina e consequentemente a diminuição do NM e; b) quando as máquinas param para atividades de MP, aumentado o NM para seu estado $NM_{j,t}^{max}$. Se uma MA é programada após uma certa tarefa, o *setup* da próxima tarefa não pode ser iniciado antes que essa MA tenha sido finalizada. Se, após o processamento de uma certa tarefa a máquina parar para realizar MA's de dois de seus componentes, ambas são realizadas sequencialmente, ou seja, uma MA após a outra. Um NM negativo está associado com a quebra de máquina, então, uma programação factível deve assegurar que o NM não ultrapasse valores abaixo de zero para evitar as indisponibilidades das máquinas. Assume-se que os tempos de *setup* e de processamento são inteiros. Cada tarefa e seus respectivos *setups* possuem uma taxa de deterioração fixa e conhecida e consistem em um fator multiplicativo das suas durações. A taxa de deterioração $\delta_{t,j,i}$ é o fator multiplicativo que descreve a parcela de deterioração que o processamento da tarefa i causa no $NM_{j,t}^{max}$. De modo análogo, $\vartheta_{j,p,i}^t$ é definido como a taxa de deterioração que o *setup* da tarefa i causa em $NM_{j,t}$. As seguintes hipóteses são consideradas para este problema:

- A duração de uma MA de um nível t na máquina j ($MA_{t,j}$) é conhecida e invariável. Isto significa que a $MA_{t,j}$ possui a mesma duração e é realizada por completo, independente do estado de deterioração da máquina e de quantas vezes é necessária;
- O valor de $NM_{t,j}$ decresce de acordo com a função linear dada por $(\delta_{t,j,i} \cdot p_{i,j} + s_{j,p,k} \cdot \vartheta_{j,p,i}^t)$;
- A $MA_{t,j}$ possui um custo total correspondente a soma do custo de substituição de peças ($SP_{t,j}$) e do custo total de mão-de-obra ($WF_t \times MA_{t,j}$);
- A quantidade $(\delta_{t,j,i} \cdot p_{i,j} + s_{j,p,k} \cdot \vartheta_{j,p,i}^t)$ é sempre menor ou igual a $NM_{j,t}^{max}$, caso contrário, o problema seria infactível;
- Antes de processar a primeira tarefa, todos os NM's de todas as máquinas estão no seu nível máximo $NM_{j,t}^{max}$;
- $NM_{j,t}^{max}$ é estabelecido por um dos cenários de MP, descritos na Seção 3.5;
- Os recursos (materiais, força de trabalho, operadores, peças para manutenção, etc.) estão sempre disponíveis para processamento das tarefas e das MA's e são ilimitados;
- Não são admitidas quebras de máquina aleatórias, ou seja, as manutenções corretivas (MC) não são consideradas neste estudo.

3.3 SCHEDULING E MANUTENÇÃO PREVENTIVA

Na prática, o problema *flow shop* com bloqueio ocorre em ambientes que não possuem ou estão impossibilitados de terem estoques intermediários entre os estágios de produção (máquinas). Alguns exemplos já estudados podem ser vistos no Capítulo 2. Para incluir restrições de tempo de *setup* e manutenção preventiva (MP), suponha-se o exemplo de uma fábrica de telhas cerâmicas esmaltadas, apresentada por Ruiz *et al.* (2005). O processo de produção da telha consiste basicamente das operações de preparação da matéria-prima (argila), extrusão, prensa de moldagem, secagem, esmaltação, queima, teste de qualidade, armazenagem, envase e expedição. Devido as restrições do processo, não existe estoque intermediário entre as operações de moldagem e secagem. Deste modo, se existe uma tarefa em processamento na operação de secagem, esta tarefa bloqueia a operação da tarefa que está na etapa de moldagem até que a secagem seja finalizada. Como exemplo da relevância do tempo de *setup* ao problema, suponha-se que a linha de produção está configurada para produzir a telha cerâmica esmaltada da cor preta com dimensões 35×35 cm e deseja-se produzir o produto

telha cerâmica esmaltada da cor branca com dimensões 45×45 cm. Um tempo de *setup* significativo será necessário para configurar os moldes de 45×45 cm na prensa e para limpeza dos equipamentos de esmaltação para adicionar o esmalte branco. No entanto, a mudança para uma telha amarela de 33×33 cm exigirá apenas a limpeza na linha de esmaltação, uma vez que a prensa de moldagem já está configurada. Logo, o tempo de *setup* é dependente da sequência. Considerando que cada etapa do processo possui máquinas com componentes reparáveis que estão sujeitas a falhas após um período de operação, podem-se estabelecer políticas de MP para os componentes das máquinas, permitindo um melhor gerenciamento dos recursos. Por exemplo, a manutenção do nível de óleo das prensas possui uma periodicidade de manutenção diferente da manutenção dos componentes do motor da prensa de moldagem. Para fins de estudo, as taxas de deterioração destes componentes podem ser representadas como funções lineares do tipo $(\delta_{t,j,i} \cdot p_{i,j} + s_{j,p,k} \cdot \vartheta_{j,p,i}^t)$, que representa a deterioração, em unidades de tempo, do nível de MP t da máquina j durante o processamento da tarefa i e de seu respectivo *setup* na máquina j . Ao atingir o nível mínimo de MP, na prática, existe uma alta probabilidade de quebra da máquina, e, teoricamente, considera-se que a máquina quebrou. As taxas de deterioração podem ser representadas pelo número de horas que as máquinas são utilizadas para realizar suas respectivas operações.

Outro exemplo de aplicação prática pode ser vista em Yu e Seif (2016), nas operações de terraplanagem. Esta atividade é altamente dependente do maquinário utilizado para realizar a terraplanagem. Os equipamentos mais comuns utilizados para o processo são escavadoras, carregadores e caminhões de transporte. O primeiro passo é a preparação do terreno que consiste em escavar a terra, operação realizada pelas escavadoras. Na próxima etapa, caminhões carregadores carregam o solo removido em caminhões transportadores. Finalmente, na etapa de transporte, os caminhões transportam a terra até o ponto de descarregamento por meio de rotas com diferentes declives e com diferentes condições de rodovias. Atividades de MP típicas para terraplanagem é geralmente baseada nas horas de serviço do maquinário. Os intervalos de manutenção determinados como o nível máximo de manutenção são estabelecidos pela empresa fabricante dos veículos, como por exemplo, os dados apresentados na Tabela 8.

Tabela 8 - Intervalos de manutenção (horas) recomendados pela empresa fabricante dos veículos

Máquina	10	50	100	250	500	1.000	2.000	3.000	4.000	5.000	6.000
Escavadoras	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
Carregadores	✓	✓	✓	✓	✓	✓	✓				
Caminhões	✓	✓			✓	✓	✓	✓	✓	✓	✓

Fonte: Adaptado de Yu e Seif (2016).

Os intervalos mostrados na Tabela 8 podem ser considerados o nível máximo de manutenção $NM_{j,t}^{max}$. Diferentes tarefas são incluídas em cada MA. Por exemplo, as tarefas inclusas nas atividades de 50 horas são lubrificação, verificação do eixo de transmissão, etc.

Na Tabela 9 apresenta-se pedidos de terraplanagem em 4 locais (tarefas) diferentes, em que as operações de terraplanagem precisam ser feitas. Há uma escavadora, um carregador e um caminhão disponíveis. Na Tabela 9 apresenta-se os dias necessários para operação das máquinas e as horas de funcionamento que a máquina pode operar no local.

Tabela 9 - Dias de operação $\times$ taxas de deterioração (horas)

Locais (tarefas)	Escavadora	Carregadora	Caminhão
A	20×5	20×5	40×3
B	14×8	14×6	13×8
C	20×4	21×5	20×5
D	30×3	40×2	30×5

Fonte: Adaptado de Yu e Seif (2016).

A quantidade de horas em que uma máquina pode trabalhar em um local pode ser diferente em outros locais devido a condição do local. Como resultado, as operações descritas na Tabela 9 são expressas como os dias de uso da máquinas multiplicado pelas horas que uma máquina pode trabalhar, que pode representar as taxas de deterioração dos NM's em horas.

As distâncias entre estes 4 locais podem variar de acordo com a sequência de processamento estabelecida pelo programador. Por exemplo, a distância entre o local A e o local B é diferente da distância entre o local A e o local C. Consequentemente, o tempo de transporte das máquinas até o local de operação pode ser representado como tempos de *setup* dependentes da sequência de atuação nos locais. Por questões de custos, a operação de remoção da terra, feita pela escavadeira, só pode iniciar uma nova tarefa quando a operação de carregamento da tarefa anterior já estiver finalizada. O mesmo ocorre entre as operações de carregamento e transporte do solo. Devido a esta condição, a tarefa atual fica bloqueada até que a máquina seguinte termine o processamento da tarefa anterior. Pode-se então considerar este problema de programação da terraplanagem como um *flow shop* com 3 máquinas e com bloqueio, tempos de *setup* dependentes da sequência e MP com níveis de manutenção.

A decisão da programação das atividades de MP consiste em determinar entre quais tarefas as atividades de MP devem ser realizadas, de modo que todas elas sejam realizadas

dentro de seus respectivos intervalos máximos $NM_{j,t}^{max}$. Como já pontuado por Cassady e Kutanoglu (2003), a integração entre MP e *scheduling* gera resultados mais eficazes que em relação a programar cada uma delas individualmente.

Um exemplo em gráfico de *gantt* do problema *flow shop* com duas máquinas, ocorrência de bloqueio, tempos de *setup* das máquinas dependentes da sequência de tarefas e ocorrências de MP pode ser visto na Figura 3.

Figura 3 - Problema *flow shop* com duas máquinas, com bloqueio, com tempos de *setup* dependentes da sequência e com ocorrências de manutenção preventiva de dois componentes reparáveis

Fonte: Autor, 2019.

Na Figura 3 podem ser vistos dois níveis de MP com diferentes intervalos de tempo, representados por N_1 e N_2 para cada máquina. O termo $B_{1,2,1}$ expressa o tempo de bloqueio da tarefa 2 na máquina 1. $B_{1,2,1}$ é causado pela atividade de manutenção (MA) do componente 1 na máquina 2 ($MA_{1,2}$) e pelo tempo de *setup* da tarefa 1 para a tarefa 2 na máquina 2 ($S_{1,2,2}$). Por causa deste tempo de bloqueio, M_1 está liberada para operar somente após J_2 ser liberada para ser processada em M_2 . Enquanto J_2 é mantida na M_1 , não é permitido realizar a MA ($MA_{1,1}$) nem o *setup* ($S_{2,3,1}$) na M_1 . Por isso, $MA_{1,1}$ e $S_{2,3,1}$ são realizadas somente no início de processamento da J_2 na M_2 .

O nível de manutenção (NM) (descritos como N_t de M_j Figura 3) decresce à medida que as tarefas e seus respectivos *setups* são processados. Considera-se que, quando o NM atinge o valor zero, uma MA é realizada para reparar a máquina ao seu nível máximo $NM_{j,t}^{max}$. Caso o

NM atinja um valor abaixo de zero, a MP é realizada antes do início de realização do *setup* da tarefa que ocasionará este valor.

Seguindo a terminologia de Graham *et al.* (1979), os problemas podem ser denotados como $F_m|block, NM_j^t, s_{j,k}|(C_{max}|total_cost = C)$ e $F_m|block, NM_j^t, s_{j,k}|(\sum_i C_i|total_cost = C)$ em que NM_j^t descreve o nível de MP do componente t na máquina j e MP_{cost} o limitante do custo total de MP.

3.3.1 Modelo matemático proposto para os problemas com minimização de duração total da programação e tempo total de fluxo

A seguir descreve-se as variáveis e parâmetros utilizados para descrever o modelo matemático proposto.

Variáveis	
$C_{[i],j}$	Data de término da tarefa que ocupa a posição i na máquina j
$R_{[i],j}$	Data de início da tarefa que ocupa a posição i na máquina j
$a_{[i],j,t}$	Nível de deterioração do nível t na máquina j antes de processar a tarefa que ocupa a posição i
$\tilde{a}_{[i],j,t}$	Nível de deterioração do nível t na máquina j após processar a tarefa que ocupa a posição i
$z_{[i],p,k}$	Variável binária que assume valor 1 se a tarefa p precede a tarefa k e 0 se p não são adjacentes k .
$x_{[i],k}$	Variável binária que assume valor 1 se a tarefa k é inserida na posição i e, 0, caso contrário.
$y_{[i],j,t}$	Variável binária que assume valor 1 se a manutenção do tipo t é realizada antes da tarefa que ocupa a posição i na máquina j
$total_cost$	Custo total de Manutenção Preventiva (MP)
Parâmetros	
$MA_{j,t}$	Duração da MP do nível t na máquina j
$s_{j,p,k}$	Tempo de <i>setup</i> entre as tarefas k e p na máquina j
$NM_{j,t}^{max}$	Nível máximo da MP do nível t na máquina j

$p_{k,j}$	Tempo de processamento da tarefa k na máquina j
$\delta_{t,j,i}$	Taxa de deterioração do tempo de processamento $p_{k,j}$ no nível t
$g_{j,p,k}^t$	Taxa de deterioração do <i>setup</i> $s_{j,p,k}$ no nível t
MP_{cost}	Limitante do custo total de MP
WF_t	Custo de mão-de-obra para MP de nível t
$SP_{t,j}$	Custo de partes e componentes para MP de nível t na máquina j
M, M' ,	Número muito grandes

Como já mencionado, dois objetivos de desempenho são considerados: duração total da programação e tempo total de fluxo, representados pelas equações (8) e (9), respectivamente.

$$\min C_{[n],m} \quad (8)$$

$$\min \sum_{i=1}^n C_{[i],m} \quad (9)$$

As restrições (10) e (11) garantem que apenas uma tarefa seja alocada em apenas uma posição da sequência e vice-versa.

$$\sum_{i=1}^n x_{[i],k} = 1 \quad k = 1, \dots, n \quad (10)$$

$$\sum_{k=1}^n x_{[i],k} = 1 \quad i = 1, \dots, n \quad (11)$$

Os tempos de *setup* seguiram, de forma análoga, as restrições descritas por Takano e Nagano (2017), ou seja,

$$\sum_{p=1}^n \sum_{k \neq p} z_{[0],p,k} = 0 \quad (12)$$

$$\sum_{p=1}^n \sum_{k \neq p} z_{[i],p,k} = 1 \quad i = 2, \dots, n \quad (13)$$

$$z_{[i],p,k} \geq x_{[i-1],p} + x_{[i],k} - 1 \quad p, k = 1, \dots, n \quad p \neq k \quad i = 2, \dots, n \quad (14)$$

A restrição (12) garante que a variável binária para o tempo de *setup* seja zero para todas as combinações de tarefas antes da tarefa que ocupa a primeira posição. A restrição (13) garante que apenas o tempo de *setup* da tarefa p , alocada na posição $i - 1$ da sequência e a tarefa k , programada na posição i , seja ativado. A restrição (14) pode assumir três valores: 1, caso a tarefa p preceda a tarefa k , 0 se apenas uma das duas variáveis assuma valor 1 e -1 se nenhuma das duas tarefas fazem parte das posições $i-1$ e i .

$$R_{[1],1} = \sum_{k=1}^n x_{[1],k} \times s_{j,k,k} \quad (15)$$

A restrição (15) garante a data de início da tarefa que ocupa a primeira posição da sequência na primeira máquina.

$$R_{[1],j} \geq \sum_{k=1}^n x_{[1],k} \times s_{j,k,k} \quad j = 2, \dots, m \quad (16)$$

$$R_{[1],j} \geq C_{[1],j-1} \quad j = 2, \dots, m \quad (17)$$

As restrições (16) e (17) garantem que a data início das operações de MP e tempos de *setup* sejam realizadas após o bloqueio ou após a data de término de alguma tarefa, respectivamente.

$$R_{[i],j} \geq R_{[i-1],j+1} + \sum_{p=1}^n \sum_{k \neq p} z_{[i],p,k} \times s_{j,p,k} + \sum_{t=1}^l y_{[i],j,t} \times MA_{j,t} \quad j = 1, \dots, m-1 \quad i = 2, \dots, n \quad (18)$$

$$R_{[i],j+1} \geq C_{[i],j} \quad j = 1, \dots, m-1 \quad i = 1, \dots, n \quad (19)$$

Para o restante das tarefas, $R_{[i],j}$ é dependente da data de início da tarefa que ocupa a posição $i - 1$ na máquina $j + 1$ ou de seu término na máquina $j - 1$. Esta decisão pode ser expressa pelas restrições (18) e (19). As restrições (20) determinam as datas de término de todas as tarefas em todas as máquinas.

$$C_{[i],j} = R_{[i],j} + \sum_{k=1}^n x_{[i],k} \times p_{k,j} \quad j = 1, \dots, m \quad i = 1, \dots, n \quad (20)$$

As decisões de alocação da MP seguiram o modelo proposto por Yu e Seif (2016). Neste sentido, as restrições (21) garantem que nenhum nível de deterioração t antes de processar a tarefa que ocupa a posição i na máquina j seja menor que a soma da quantidade deteriorada que o tempo de processamento e seu respectivo tempo de *setup* causa em $NM_{t,j}$.

$$a_{[i]j,t} \geq \sum_{k=1}^n x_{[i],k} \times p_{k,j} \times \delta_{t,j,k} + \sum_{p=1}^n \sum_{k \neq p}^n z_{[i],p,k} \times s_{j,p,k} \times 9_{j,p,k}^t \quad \begin{matrix} t = 1, \dots, l & j = 1, \dots, m \\ i = 2, \dots, n \end{matrix} \quad (21)$$

Para atender as hipóteses de MP descritas na Seção 3.2, as restrições (22) garantem que o nível de deterioração t da máquina j antes de processar a primeira tarefa esteja em seu nível máximo.

$$a_{[1]j,t} = NM_{j,t}^{max} \quad t = 1, \dots, l \quad j = 1, \dots, m \quad (22)$$

As restrições (23) determinam o nível de deterioração da MP de nível t após o processamento da primeira tarefa. O nível de deterioração da MP de nível t na máquina j após o processamento do restante das tarefas é expresso pelas restrições (24).

$$\tilde{a}_{[1]j,t} = a_{[1]j,t} - \sum_{k=1}^n x_{[1],k} (p_{k,j} \times \delta_{t,j,k} + s_{j,k,k} \times 9_{j,k,k}^t) \quad t = 1, \dots, l \quad j = 1, \dots, m \quad (23)$$

$$\tilde{a}_{[i]j,t} = a_{[i]j,t} - \left(\sum_{k=1}^n x_{[1],k} \times p_{k,j} \times \delta_{t,j,k} + \sum_{p=1}^n \sum_{k \neq p}^n z_{[1],p,k} \times s_{j,k,k} \times 9_{j,k,k}^t \right) \quad \begin{matrix} t = 1, \dots, l \\ j = 1, \dots, m \end{matrix} \quad (24)$$

A decisão de se realizar uma MP foi focada na variável $a_{[i]j,t}$. Neste sentido a variável binária $y_{[i]j,t}$ assumirá valor 1 se uma operação de MP é realizada e 0, caso contrário. Esta condição é definida pelas restrições (25):

$$a_{[i]j,t} = \begin{cases} \tilde{a}_{[i-1]j,t} & y_{[i]j,t} = 0 \\ NM_{j,t}^{max} & y_{[i]j,t} = 1 \end{cases} \quad t = 1, \dots, l \quad j = 1, \dots, m \quad i = 2, \dots, n \quad (25)$$

Para linearizar as restrições (25), utilizou-se a técnica do “ M Grande”, gerando as restrições (26), (27), (28) e (29):

$$a_{[i]j,t} > \tilde{a}_{[i-1]j,t} - My_{[i]j,t} \quad t = 1, \dots, l \quad j = 1, \dots, m \quad (26)$$

$$a_{[i]j,t} < \tilde{a}_{[i-1]j,t} + My_{[i]j,t} \quad t = 1, \dots, l \quad j = 1, \dots, m \quad (27)$$

$$a_{[i]j,t} > NM_{j,t}^{max} - M' (1 - y_{[i]j,t}) \quad t = 1, \dots, l \quad j = 1, \dots, m \quad (28)$$

$$a_{[i]j,t} < NM_{j,t}^{max} + M' (1 - y_{[i]j,t}) \quad t = 1, \dots, l \quad j = 1, \dots, m \quad (29)$$

É interessante observar que, se $y_{[i]j,t} = 1$, as restrições (26) e (27) são ativadas, e $a_{[i]j,t}$ recebe o valor $NM_{j,t}^{max}$, sem que as restrições (28) e (29) sejam invalidadas. De modo análogo, se $y_{[i]j,t} = 0$, as restrições (28) e (29) são ativadas e $a_{[i]j,t}$ recebe o valor de $\tilde{a}_{[i-1]j,t}$, sem que as restrições (26) e (27) sejam invalidadas.

Os custos totais podem ser expressos de acordo com a restrição (30). A restrição (31) garante que o custo total de MP encontrado seja menor ou igual que o parâmetro do custo total de MP dado.

$$total_cost = \sum_{t=1}^l \sum_{j=1}^m \sum_{i=2}^n y_{[i]j,t} \times (WF_t \times MA_{j,t} + SP_{t,j}) \quad (30)$$

$$total_cost \leq MP_{cost} \quad (31)$$

Por fim, as restrições (32) e (33) descrevem o tipo das variáveis de decisão.

$$x_{[i]k}, z_{[i]p,k}, y_{[i]j,t} \in \{0,1\} \quad t = 1, \dots, l \quad j = 1, \dots, m \quad i = 1, \dots, n \quad (32)$$

$$total_cost, a_{[i]j,t}, \tilde{a}_{[i]j,t}, C_{[i]j}, R_{[i]j} \geq 0 \quad t = 1, \dots, l \quad j = 1, \dots, m \quad i = 1, \dots, n \quad (33)$$

3.3.2 Cálculo das funções-objetivo

Para identificar os fatores que influenciam o cálculo das datas de início, de término e de partida das tarefas em todas as máquinas, apresenta-se na Figura 4 uma programação viável

para os problemas $F_m | block, NM_j^t, s_{j,k} | (\sum_i C_i | MP_{cost} = C)$ ou $F_m | block, NM_j^t, s_{j,k} | (C_{max} | MP_{cost} = C)$.

Figura 4 - Flow shop com 5 tarefas e com 4 máquinas

Fonte: Autor, 2019.

Pela Figura 4 pode-se perceber que a data de término da tarefa J_1 na máquina M_2 ($C_{1,2}$) é dada por $C_{1,2} = C_{1,2} + p_{1,2} = S_{1,1} + p_{1,1} + p_{1,2}$. Porém, convém salientar que se caso $S_{1,2}$ seja maior que $C_{1,1}$, ou seja, a data de término da J_1 na máquina M_1 , $C_{1,2}$ é obtido por $C_{1,2} = S_{1,2} + p_{1,2}$. As mesmas situações podem ocorrer para o restante das máquinas. Para generalizar o cálculo das datas de término da tarefa que ocupa a primeira posição da sequência denota-se $R_{[i],j}$, $C_{[i],j}$, $p_{[i],j}$ e $S_{[i],j}$ como as datas de início, de término, o tempo de processamento e o tempo de *setup* da tarefa que ocupa a posição i na máquina j , respectivamente. Pode-se determinar o valor de $C_{i,j}$ por meio do cálculo da $R_{i,j}$ de todas as tarefas. A $R_{i,j}$ é dada pela data em que a M_j está liberada para processar J_i . A data de início da primeira tarefa da sequência em qualquer máquina pode ser definida como $R_{1,j} = \max(C_{1,j-1}; S_{j,1})$ para $j = 1, \dots, m$, considerando $C_{1,0} = 0$. Por fim, a data de término da primeira tarefa da sequência na máquina j ($C_{1,j}$) pode ser facilmente calculada por meio de $C_{1,j} = R_{1,j} + p_{1,j}$ para $j = 1, \dots, m$.

Da segunda tarefa até a última, torna-se necessário, além do tempo de *setup*, incluir a ocorrência das atividades de MP de cada componente t na máquina j . Pode-se observar pela Figura 4 que as datas de início das tarefas são dadas por $R_{[i],1} = C_{[i-1],1} + \sum_{t=1}^l MA_{[i-1],[i],1}^t + S_{[i-1],[i],1}$ ou $R_{[i],1} = R_{[i-1],2} + \sum_{t=1}^l MA_{[i-1],[i],1}^t + S_{[i-1],[i],1}$. Então, a data de início de qualquer tarefa que ocupa a posição i na M_1 , $R_{[i],1}$ é dada por $R_{[i],1} = \max(C_{[i-1],1}; R_{[i-1],2}) + \sum_{t=1}^l MA_{[i-1],[i],1}^t + S_{[i-1],[i],1}$.

Da segunda e terceira máquina, M_2 e M_3 respectivamente, pode-se constatar que também há duas possibilidades de cálculo da data de início das tarefas. A primeira é $R_{[i],j} = R_{[i-1],j+1} + \sum_{t=1}^l MA_{[i-1],[i],j}^t + S_{[i-1],[i],j}$ como pode ser visto em $R_{2,2}$, $R_{2,3}$, $R_{4,2}$ e $R_{4,3}$ e $R_{5,3}$. A segunda é representada por $R_{[i],j} = C_{[i],j-1}$, como em $R_{2,3}$, $R_{3,2}$, $R_{3,3}$ e $R_{5,2}$. Logo, o cálculo de $R_{[i],j}$ para $j = 2, \dots, m-1$ pode ser obtida por $R_{[i],j} = \max(C_{[i],j-1} ; R_{[i-1],j+1} + \sum_{t=1}^l MA_{[i-1],[i],j}^t + S_{[i-1],[i],j})$

Para a última máquina (M_4) é observado que além da data de término da tarefa i na máquina M_3 , a data de início $R_{i,4}$ pode também ser composta por $R_{[i],4} = C_{[i-1],4} + \sum_{t=1}^l MA_{[i-1],[i],4}^t + S_{[i-1],[i],4}$. Tem-se então que $R_{[i],m} = \max(C_{[i],j-1} ; C_{[i-1],j+1} + \sum_{t=1}^l MA_{[i-1],[i],j}^t + S_{[i-1],[i],j})$. A data de término de todas as tarefas em todas as máquinas pode ser determinada por $C_{[i],j} = R_{[i],j} + p_{[i],j}$. Por fim, uma sumarização do cálculo das datas de início, de término e de partida das tarefas são listadas e enumeradas conforme segue:

$$C_{1,0} = 0 \quad (34)$$

$$R_{[1],j} = \max(C_{[1],j-1} ; S_{[1],j}) \quad j = 1, \dots, m \quad (35)$$

$$R_{[i],1} = \max(C_{[i-1],1} ; R_{[i-1],2}) + \sum_{t=1}^l MA_{[i-1],[i],1}^t + S_{[i-1],[i],1} \quad i = 2, \dots, n \quad (36)$$

$$R_{[i],j} = \max(C_{[i],j-1} ; R_{[i-1],j+1} + \sum_{t=1}^l MA_{[i-1],[i],j}^t + S_{[i-1],[i],j}) \quad j = 2, \dots, m-1 \quad (37)$$

$$i = 2, \dots, n$$

$$R_{[i],m} = \max(C_{[i],j-1} ; C_{[i-1],j+1} + \sum_{t=1}^l MA_{[i-1],[i],j}^t + S_{[i-1],[i],j}) \quad i = 2, \dots, n \quad (38)$$

$$C_{[i],j} = R_{[i],j} + p_{[i],j} \quad i = 1, \dots, n \quad (39)$$

$$j = 1, \dots, m$$

$$D_{[1],0} = \max(C_{[1],j-1} ; S_{[1],j+1}) \quad (40)$$

$$D_{[1],j} = \max(D_{[1],j-1} ; S_{[1],j+1}) \quad j = 2, \dots, m-1 \quad (41)$$

$$D_{[i],1} = R_{[i-1],2} \quad i = 2, \dots, n \quad (42)$$

$$D_{[i],j} = \max(C_{[i],j-1} ; R_{[i-1],j+1}) \quad j = 2, \dots, m-1 \quad (43)$$

$$i = 2, \dots, n$$

$$D_{[i],m} = C_{[i],m} \quad i = 1, \dots, n \quad (44)$$

3.4 BUSCA LOCAL PARA PROGRAMAÇÃO E REPOSICIONAMENTO DAS MANUTENÇÕES PREVENTIVAS PARA MINIMIZAÇÃO DA FUNÇÃO-OBJETIVO

Uma forma de calcular a ocorrência das atividades de manutenção preventiva (MA's) a partir da sequência gerada, dado que seja encontrado o número mínimo de MP's para um nível t na máquina j ($NM_{t,j}$), consiste em seguir o critério estabelecido por Ruiz et al. (2007): A cada vez que uma tarefa é processada, o nível de deterioração da MP t na máquina j após o processamento da tarefa i ($\tilde{a}_{[i],j,t}$) é decrescida pela quantidade $(p_{i,j} \times \delta_{t,j,i} + s_{j,k,i} \times 9_{j,k,i}^t)$. Caso $\tilde{a}_{[i],j,t}$ seja menor que zero, uma atividade de manutenção preventiva é realizada antes do processamento da tarefa que ocupa a posição i , ou seja, uma $AM_{t,j}$ é realizada antes da operação de *setup* da tarefa i e $a_{[i],j,t} = NM_{t,j}^{max}$. Consequentemente, o custo total de MP (*total_cost*) é acrescido com os custos de mão-de-obra e de substituição de peças. Utilizando-se as notações descritas na Subseção 3.3, apresenta-se na Figura 5, o pseudocódigo do procedimento de Ruiz et al. (2007).

```

//Procedimento Prog_PM_Ruiz
 $\pi \leftarrow$  Sequência de tarefas
 $MA_{[i-1],[i],j}^t(\pi) \leftarrow 0$  e posição_MP $_{t,j,[i]} \leftarrow 0$  para  $t = 1, \dots, l; j = 1, \dots, m; i = 2, \dots, n$ 
total_cost  $\leftarrow 0$ 
Para  $t \leftarrow 1$  até  $l$  faça
  Para  $j \leftarrow 1$  até  $m$  faça
     $a_{[1],j,t} \leftarrow NM_{t,j}^{max}$ 
     $\tilde{a}_{[1],j,t} \leftarrow a_{[1],j,t} - (p_{[1],j} \times \delta_{t,j,[1]} + s_{j,[1],[1]} \times 9_{j,[1],[1]}^t)$ 
    Para  $i \leftarrow 2$  até  $n$  faça
       $a_{[i],j,t} \leftarrow \tilde{a}_{[i-1],j,t}$ 
       $\tilde{a}_{[i],j,t} \leftarrow a_{[i],j,t} - (p_{[i],j} \times \delta_{t,j,[i]} + s_{j,[i-1],[i]} \times 9_{j,[i-1],[i]}^t)$ 
      Se  $\tilde{a}_{[i],j,t} < 0$  então
         $a_{[i],j,t} \leftarrow NM_{t,j}^{max}$ 
         $\tilde{a}_{[i],j,t} \leftarrow a_{[i],j,t} - (p_{[i],j} \times \delta_{t,j,[i]} + s_{j,[i-1],[i]} \times 9_{j,[i-1],[i]}^t)$ 
         $MA_{[i-1],[i],j}^t(\pi) \leftarrow MA_{t,j}$ 
        posição_MP $_{t,j,[i]} \leftarrow 1$ 
        total_cost  $\leftarrow$  total_cost + ( $WF_t \times MA_{j,t} + SP_{t,j}$ )
      FimSe
    FimPara  $i$ 
  FimPara  $j$ 
FimPara  $t$ 
Retorne  $MA_{[i-1],[i],j}^t(\pi)$  e total_cost( $\pi$ )
Fim // Prog_PM_Ruiz

```

Figura 5 - Pseudocódigo do procedimento de posicionamento de MP's *Prog_PM_Ruiz*

Fonte: Adaptado de Ruiz et al. (2007).

Neste sentido, tendo-se a matriz de atividades de MP (MA) $MA_{[i-1],[i],j}^t$, que consiste em determinar a duração das MP's do nível t entre tarefas adjacentes na máquina j , é possível calcular as datas de início, término e partida das tarefas, representados pelas Eqs. (34)) – ((43).

A variável $posição_MP_{t,j,[i]}$ recebe o valor 1 se uma MP do nível t na máquina j é realizada imediatamente antes da tarefa que ocupa a posição i da sequência e a variável $MA_{[i-1],[i],j}^t$ corresponde a duração da MP do nível t entre as tarefas que ocupam as posições $i-1$ e i na máquina j .

O procedimento descrito na Figura 5 fornece o número mínimo de atividades de MP's ou o custo total de MP mínimo que devem ser realizadas para uma dada sequência π , uma vez que programa as MA's somente quando necessário. Entretanto, o critério estabelecido por Ruiz et al. (2007) não avalia outras posições possíveis da sequência de tarefas em que as MA's podem ser posicionadas. Neste sentido, um procedimento de verificação e reposicionamento das MA's em outras posições, respeitando-se o número de MA's estabelecidas pelo critério de Ruiz et al. (2007) é proposto. O procedimento necessita como dado de entrada a sequência pré-estabelecida por *Prog_PM_Ruiz*, e tem como objetivo avaliar, a partir das posições fixadas de cada MA, a possibilidade de reposicioná-la em outras posições avaliando se há minimização da função-objetivo, dado que o custo total de MP seja mantido, ou seja, nenhuma outra MA seja realizada além das que já foram programadas. Deste modo, o procedimento proposto tem como objetivo melhorar uma solução já obtida por *Prog_PM_Ruiz*, sendo utilizada como segunda fase para o problema de minimização de C_{max} ou de $\sum_i C_i$. O pseudocódigo do procedimento mostrado na Figura 6 tem por objetivo realizar a verificação da viabilidade de se reposicionar as MA's em outras posições, a partir da última MA programada até a primeira, levando-se em consideração o monitoramento dos NM de cada MA.

```

//Procedimento Posicionamento_MP
 $\pi \leftarrow$  Sequência de tarefas
 $MA_{[i-1],[i],j}^t \leftarrow Prog\_PM\_Ruiz(\pi)$ ;  $MA_{[i-1],[i],j}^t \leftarrow MA_{[i-1],[i],j}^t$ ;  $aux\_FO \leftarrow FO(\pi)$ 
 $posição\_MP_{t,j,[i]} \leftarrow Prog\_PM\_Ruiz(\pi)$ ;  $posição\_MP_{t,j,[h]} \leftarrow posição\_MP_{t,j,[i]}$ 
 $a_{[1],j,t} \leftarrow NM_{t,j}^{max}$ 
 $\tilde{a}_{[1],j,t} \leftarrow a_{[1],j,t} - (p_{[n],j} \times \delta_{t,j,[n]} + s_{j,[n-1],[n]} \times g_{j,[n-1],[n]}^t)$ 
 $h \leftarrow n-1$ ;  $r \leftarrow 2$ ;
se  $posição\_MP_{t,j,[h]} \neq 0$  então
     $MA_{[h-1],[h],j}^t \leftarrow MA_{[h-1],[h],j}^t - MA_{j,t}$ 
     $k \leftarrow n$ 
FimSe
Enquanto  $h > -1$  faça
    Se  $posição\_MP_{t,j,[h]} \neq 0$  então
         $MA_{[h-1],[h],j}^t \leftarrow MA_{[h-1],[h],j}^t - MA_{j,t}$ 
         $k \leftarrow h$ 
    FimSe
     $a_{[r],j,t} \leftarrow \tilde{a}_{[r-1],j,t}$ 
     $\tilde{a}_{[r],j,t} \leftarrow a_{[r],j,t} - (p_{[h],j} \times \delta_{t,j,[h]} + s_{j,[h-1],[h]} \times g_{j,[h-1],[h]}^t)$ 
    Se  $\tilde{a}_{[r],j,t} < 0$  então
         $a \leftarrow k$ 
        Se  $h < k$  então
            Para  $b \leftarrow k$  até  $h+1$  faça
                 $MA_{[h-1],[h],j}^t \leftarrow MA_{[h-1],[h],j}^t + MA_{j,t}$ 
                Se  $aux\_FO \geq FO(MA_{[h-1],[h],j}^t)$  então
                     $aux\_FO \leftarrow FO(MA_{[h-1],[h],j}^t)$ 
                     $a \leftarrow b$ 
                FimSe
                 $MA_{[h-1],[h],j}^t \leftarrow MA_{[h-1],[h],j}^t - MA_{j,t}$ 
            FimPara
             $MA_{[a-1],[a],j}^t \leftarrow MA_{[a-1],[a],j}^t + MA_{j,t}$ 
            Se  $a \neq k$  então
                 $posição\_MP_{t,j,[a]} \leftarrow 1$ 
                 $posição\_MP_{t,j,[k]} \leftarrow 0$ 
                 $\tilde{a}_{[1],j,t} \leftarrow NM_{t,j}^{max}$ 
                 $r \leftarrow 1$ ;  $h \leftarrow a$ 
            Senão
                 $h \leftarrow -1$ 
            FimSenão
        Senão
             $h \leftarrow -1$ 
        FimSenão
    FimSe
     $h \leftarrow h - 1$ 
     $r \leftarrow r + 1$ 
FimEnquanto  $h$ 
Fim//Procedimento Posicionamento_MP

```

Figura 6 - Pseudocódigo do procedimento de avaliação da posição das manutenções preventivas

Fonte: Autor, 2019.

De modo geral, o procedimento descrito na Figura 6 realiza uma busca do fim da sequência para o início da sequência, tentando reposicionar, a partir da última MP do nível t na

máquina j , a posição entre tarefas que minimiza a função objetivo dado o nível de deterioração da MP de nível t na máquina j . O procedimento *Posicionamento_MP* é repetido para todas MA's de todos os níveis de MP em todas as máquinas.

Esta busca local leva em conta o nível de deterioração da máquina antes e depois de processar a tarefa que ocupa a i -ésima posição da sequência. Inicialmente, o procedimento calcula o tempo de operação da máquina, representados pelas variáveis $\tilde{a}_{[r]j,t}$ e $a_{[r]j,t}$, a partir da última tarefa até a h -ésima tarefa que causa $\tilde{a}_{[r]j,t} < 0$. Se a posição h se encontra antes da posição k , onde a última MP está programada, a busca local testa todas as b posições entre k e h , avaliando a função-objetivo. Caso o procedimento encontre um valor da função-objetivo menor ou igual o da programação atual em uma posição a , entre k e h , a atividade MP é reprogramada na posição a . Então, a posição $a - 1$ se torna k , o nível de deterioração da MP de nível t na máquina j é reiniciado ao seu máximo e o cálculo de deterioração recomeça a partir da tarefa que ocupa a posição $a-1$. Se $\tilde{a}_{[r]j,t} < 0$ em uma posição h anterior à da penúltima MP programada pelo procedimento *Prog_MP_Ruiz*, a busca local testa se há minimização da função-objetivo de k à h . Caso haja, a busca local reposiciona a penúltima MP na posição a entre k e h . O teste de reinserção das MP's continua até que todas as MP's sejam reposicionadas ou se uma determinada MP não apresentar valor da função-objetivo menor ou igual nas posições testadas. Na Figura 7 é mostrado o procedimento final de reposicionamento das atividades de MP.

```

//MP_Reposicionamento
 $\pi \leftarrow$  Sequência de tarefas
 $MA_{[i-1],[l],j}^t \leftarrow$  Prog_PM_Ruiz();  $MA_{[i-1],[l],j}^t \leftarrow MA_{[i-1],[l],j}^t$ ;  $best\_FO, aux\_FO \leftarrow C_{max}(\pi)$ ;  $aux\_FO'' \leftarrow$  Big M
 $posição\_MP_{t,j,[l]} \leftarrow$  Prog_PM_Ruiz( $\pi$ );  $posição\_MP_{t,j,[h]} \leftarrow$   $posição\_MP_{t,j,[l]}$ 
Enquanto  $flag \leftarrow True$  faça
  Para  $t \leftarrow 1$  até  $l$  faça
    Para  $j \leftarrow 1$  até  $m$  faça
      Aplicar Procedimento Posicionamento_PM
      Se  $aux\_FO'' > best\_FO$  então
         $aux\_FO'' \leftarrow best\_FO$ 
         $best\_MP_{t,j,[h]} \leftarrow$   $posição\_MP_{t,j,[l]}$ 
         $best\_MA_{[i-1],[l],j}^t \leftarrow MA_{[i-1],[l],j}^t$ 
      FimSe
       $posição\_MP_{t,j,[l]} \leftarrow$   $posição\_MP_{t,j,[l]}$ 
       $MA_{[i-1],[l],j}^t \leftarrow MA_{[i-1],[l],j}^t$ 
       $best\_C_{max} \leftarrow C_{max}(\pi)$ 
    FimPara  $j$ 
  FimPara  $t$ 
  Se  $FO(\pi) > aux\_FO''$  então
     $FO(\pi) \leftarrow aux\_FO''$ 
     $posição\_MP_{t,j,[l]} \leftarrow best\_MP_{t,j,[h]}$ 
     $MA_{[i-1],[l],j}^t \leftarrow best\_MA_{[i-1],[l],j}^t$ 
     $posição\_MP_{t,j,[l]} \leftarrow 0$ 
     $MA_{[i-1],[l],j}^t \leftarrow 0$ 
     $best\_C_{max} \leftarrow C_{max}(\pi)$ 
  Senão
     $flag \leftarrow False$ 
  fimSenão
FimEnquanto
Retorne  $FO(\pi)$ 
Fim //MP_Reposicionamento

```

Figura 7 - Pseudocódigo do procedimento de reposicionamento das atividades de manutenção preventiva

Fonte: Autor, 2019.

O procedimento *MP_Reposicionamento* avalia a função-objetivo ao realocar as atividades de MP de cada nível t em cada máquina j em posições anteriores a da fornecida por *Prog_PM_Ruiz*. A reprogramação que gera o menor valor da função-objetivo das $l \times m$ combinações é adotada. Se a função-objetivo for menor que o valor da sequência dada por *Prog_PM_Ruiz*, a solução corrente é substituída pela nova. Na iteração seguinte, o procedimento reinicia a busca para as $l \times m$ combinações e a melhor solução encontrada substitui a atual caso seja menor. O procedimento continua até que não haja mais melhorias na solução. Apesar de ser uma busca simples, a complexidade do algoritmo pode chegar a $O(K(lmn))$, sendo K o número de iterações até que $flag \leftarrow false$.

3.5 INTERVALOS ENTRE MANUTENÇÕES PREVENTIVAS

Um dos parâmetros necessários para a decisão de programação das manutenções preventivas (MP's) é o estabelecimento do intervalo entre MP's, que é dado pela medida máxima da MP de nível t ($NM_{j,t}^{max}$). No intuito de melhor avaliar o desempenho dos métodos de solução adotados para o problema definido, dois cenários de MP foram considerados.

O primeiro cenário, denominado cenário YS, segue as configurações determinadas por Yu e Seif (2016). Os autores estabelecem um intervalo entre MP's dado pela multiplicação da taxa de deterioração máxima do tempo de processamento e o maior tempo de processamento encontrado na instância do problema, dado por

$$NM_t^{max} = \max(\delta_{t,j,i}) \times \max(p_{i,j})$$

Devido à adição dos tempos de *setup* ao problema, a equação de Yu e Seif (2016) foi acrescida da máxima taxa deterioração do *setup* para o nível t encontrada entre todas as máquinas e o maior tempo de *setup* encontrado entre todas as máquinas da instância do problema.

$$NM_t^{max} = \max(g_{j,i,k}^t) \times \max(s_{j,i,k}) + \max(\delta_{t,j,i}) \times \max(p_{i,j}) \quad (45)$$

O segundo cenário, denominado cenário RGM, segue a política de MP determinada por Ruiz et al. (2007). Esta política consiste em realizar manutenções preventivas (MP) sistemáticas após um intervalo NM_t^{max} para assegurar uma confiabilidade mínima igual no tempo $t = 0$, ou seja, a MP restaura as máquinas a um estado “tão bom como novo”. Neste caso, as MP's são realizadas em intervalos regulares $0, 1T_{MP}, 2T_{MP}, 3T_{MP}, \dots, nT_{MP}$, considerados como pontos de renovação. Os autores utilizam a teoria estatística da confiabilidade com aplicação da função de densidade de probabilidade de Weibull para estabelecer o valor de NM_t^{max} e, posteriormente, sistematizá-la como uma política de ações de MP nas máquinas. Os conceitos básicos da distribuição de Weibull e da teoria estatística da confiabilidade podem ser vistas em Ruiz et al. (2007). Partindo-se da função de risco de Weibull, os autores chegam na seguinte política:

$$NM_t^{max} = \left[-\theta^\beta \frac{\ln R_0(t)}{t} \right]^{1/(\beta - 1)} \quad (46)$$

Onde:

θ – Parâmetro escala;

β – Parâmetro forma;

$R_0(t)$ – Função confiabilidade da máquina no tempo t ;

t – Tempo de operação estabelecido pelo tomador de decisão

Suponha que o tempo até a falha de uma máquina siga o modelo de Weibull $X \approx [\theta, \beta]$, onde $\theta = 1200$ e $\beta = 2$. O objetivo é determinar o intervalo T_{MP} , para que a confiabilidade após $t = 670$ h seja de $R(t) = 90\%$. Aplicando-se a Eq. (46), tem-se que $T_{MP} = 226,45$ horas, que significa que para cada aproximadamente 227 horas de operação da máquina, uma atividade de MP é realizada.

3.6 CONFIGURAÇÕES DAS INSTÂNCIAS DE PROBLEMAS PARA EXPERIMENTOS COMPUTACIONAIS

Foram geradas no total 6 bases de dados, considerando os cenários de MP's de Yu e Seif (2016) e a política de Ruiz *et al.* (2007). Duas bases de instâncias de problemas foram geradas para aplicação do modelo matemático proposto, outras duas para problemas de grande porte e as duas últimas foram geradas para ajuste dos parâmetros das metaheurísticas adaptadas.

Para gerar as instâncias de problemas de grande porte, utilizou-se como parâmetros o número de tarefas (n), número de máquinas (m), níveis de manutenção preventiva (l) e duração dos tempos de *setup* ($s_{j,i,k}$). Foi adotado como base para n e m , as configurações sugeridas por Ruiz *et al.* (2007) para problemas *flow shop* permutacional sujeito a políticas de manutenção preventiva (MP). O banco de problemas consiste de instâncias configuradas entre $n = \{20, 50, 100, 200, 500\}$ e $m = \{5, 10, 15, 20\}$. Os tempos de processamento das tarefas foram gerados aleatoriamente em uma distribuição uniforme no intervalo $U[1,99]$.

A inclusão dos tempos de *setup* dependentes da sequência seguiu as configurações sugeridas por Ruiz *et al.* (2005). Os autores desenvolveram quatro diferentes cenários com taxas

de *setup* diferentes a partir das instâncias de Taillard (1993). Os tempos de *setup* foram definidos como 25%, 50%, 100% e 125% do valor dos tempos de processamento, ou seja, o primeiro cenário é composto por tempos de *setup* dentro da faixa de [1, 24], o segundo cenário na faixa de [1, 49], o terceiro na faixa de [1, 99] e o último cenário na faixa de [1, 125]. Para todas os cenários, os valores dos tempos de *setup* foram gerados aleatoriamente com distribuição uniforme.

Os níveis de manutenção preventiva l e as taxas de deterioração das tarefas seguiram as mesmas configurações de Yu e Seif (2016), ou seja, $l = 1, 2, 3$ e 5 . Cada taxa de deterioração recebe um valor real aleatório entre $[0,1, 2]$ com distribuição uniforme.

Na Tabela 10 apresenta-se a sumarização das configurações dos parâmetros comuns para todas as instâncias de problemas geradas.

Tabela 10 - Configuração dos parâmetros do banco de problemas (por problema)

Parâmetros	Tamanho	Amplitude	Geração dos valores
Tempos de processamento	$n \times m$	[1, 99]	Aleatório(inteiro, com distribuição uniforme)
Tempos de <i>setup</i>	$n \times n \times m$	[1, 9], [1, 49], [1, 99] ou [1, 124]	Aleatório(inteiro, com distribuição uniforme)
Duração das MA's	l	[1, 99]	Aleatório(inteiro, com distribuição uniforme)
Taxas de deterioração da tarefa	$n \times m \times l$	[0,1, 2]	Aleatório(real, com distribuição uniforme)
Taxas de deterioração do <i>setup</i>	$n^2 \times m \times l$	[0,1, 2]	Aleatório (real, com distribuição uniforme)

Fonte: Autor, 2019.

A política proposta por Ruiz et al. (2007) necessita dos parâmetros β , θ e t . Os valores de θ seguiram as mesmas configurações estabelecidas pelos autores, ou seja, para $n = \{20, 50, 100, 200, 500\}$ tem-se que $\theta_n = \{1500, 2200, 3200, 4500, 7200\}$, respectivamente. Entretanto, é importante ressaltar que com a inclusão dos tempos de *setup*, o valor de NM_t^{max} pode não ser factível para o problema. Neste sentido, o valor do parâmetro escala (θ) para todas as combinações de n e $s_{j,i,k}$ foi calculado como

$$\theta_{n,F} = \theta_n \times F_s + \theta_n \quad (47)$$

Onde

$\theta_{n,F}$ - Parâmetro escala para $n = \{20, 50, 100, 200, 500\}$ e tempos de *setup* com fator $F_s = \{0,25; 0,5; 1; 1,25\}$;

θ_n - Valor do parâmetro escala, em que $\theta_n = \{1500, 2200, 3200, 4500, 7200\}$ para $n = \{20, 50, 100, 200, 500\}$, respectivamente;

F_s - Fator multiplicador, em que $F_s = \{0,25; 0,5; 1; 1,25\}$ para tempos de *setup* proporcionais a 25%, 50%, 100% e 125% dos tempos e processamento, respectivamente.

Na Tabela 11, são apresentados todos os valores gerados para o parâmetro θ .

Tabela 11 - Valores do parâmetro escala para problemas de grande porte

	$\theta = 1500$ para $n = 20$	$\theta = 2200$ para $n = 50$	$\theta = 3200$ para n $= 100$	$\theta = 4500$ para n $= 200$	$\theta = 7200$ para n $= 500$
$F_s = 0,25$ para $s_{j,i,k}$ $= 25\%$	1.875	2.750	4.000	5.625	9.000
$F_s = 0,50$ para $s_{j,i,k}$ $= 50\%$	2.250	3.300	4.800	6.750	10.800
$F_s = 1$ para $s_{j,i,k} =$ 100%	3.000	4.400	6.400	9.000	14.400
$F_s = 1,25$ para $s_{j,i,k}$ $= 125\%$	3.375	4.950	7.200	10.125	16.200

Fonte: Autor, 2019.

Cada nível de MP foi atribuído com um valor diferente de β . Uma análise experimental foi conduzida para determinar o valor mínimo que β pode assumir de modo que a taxa de deterioração máxima $\left(\delta_{i,j,t} \times p_{i,j} + g_{j,i,k}^t \times s_{j,i,k} = 1,9 \times 99 + 1,9 \times s_{j,i,k}\right)$ seja atendida pelo NM_t^{max} . Chegou-se ao valor de $\beta = 2,3$ e, a partir daí, β foi proposito de modo que cada nível seja contemplado com um valor de NM_t^{max} maior que o anterior. Logo, para $l = 1, l = 2, l = 3$ e $l = 5$ tem-se que $\beta = [2,3], [2,3, 2,5], [2,3, 2,5, 3]$ e $[2,3, 2,5, 3, 3,5]$ e $[4,5]$, respectivamente. Por fim, $t = n \times s_{j,i,k}/2 + n \times 50$. Para cada configuração de n, m, l , e $s_{j,i,k}$ foram gerados 10 problemas diferentes, resultando em $5 \times 4 \times 4 \times 4 \times 10 = 3.200$ problemas por cenário de MP. Portanto, 6.400 problemas de grande porte foram avaliados.

Para as instâncias de pequeno porte considerou-se $n = \{5,6,7,8\}$, $m = \{2,3,4,5\}$, $l = \{1,2,3,4\}$ e $s_{j,i,k} = \{25\%, 50\%, 100\% \text{ e } 125\%\}$. Partindo-se do mesmo princípio utilizado para

gerar os intervalos entre MP's em problemas de grande porte, os valores de θ podem ser vistos na Tabela 12.

Tabela 12 - Valores do parâmetro escala para problemas de pequeno porte

	$\theta = 950 \text{ p/ } n = 5$	$\theta = 1000 \text{ p/ } n = 6$	$\theta = 1100 \text{ p/ } n = 7$	$\theta = 1200 \text{ p/ } n = 8$
$F_s = 0,25$ para $s_{j,i,k} = 25\%$	1.187	1.250	1.375	1.500
$F_s = 0,50$ para $s_{j,i,k} = 50\%$	1.425	1.500	1.650	1.800
$F_s = 1$ para $s_{j,i,k} = 100\%$	1.900	2.000	2.200	2.400
$F_s = 1,25$ para $s_{j,i,k} = 125\%$	2.137	2.250	2.475	2.700

Fonte: Autor, 2019.

Os valores do parâmetro β foram ajustados em $[2,1]$, $[2,1, 2,15]$, $[2,1, 2,15, 2,2]$ e $[2,1, 2,15, 2,2, 2,25]$ para $l = 1, 2, 3$ e 4 , respectivamente, e $t = n \times s_{j,i,k}/2 + n \times 50$. Outros dados do problema seguiram as configurações apresentadas na Tabela 10. Para cada combinação n, m, l , e $s_{j,i,k}$ foram gerados 10 problemas, resultando em 2.560 problemas de pequeno porte para cada base, o que totaliza 5.120 problemas.

As bases de dados utilizadas para ajustamento de parâmetros das metaheurísticas seguiram as configurações propostas por Shao et al. (2018b). Logo, $n = \{40, 80, 120, 160, 200\}$, $m = \{5, 10, 15, 20\}$, $l = \{1, 2, 3, 5\}$ e $s_{j,i,k} = \{25\%, 50\%, 100\% \text{ e } 125\%\}$. O parâmetro θ foi determinado como $\{2.000, 2.800, 3.600, 4.000, 4.500\}$ para $n = \{40, 80, 120, 160, 200\}$, respectivamente. Logo, as seguintes configurações descritas na Tabela 13 foram encontradas para θ .

Tabela 13 - Valores do parâmetro escala para base de dados para ajuste de parâmetros

	$\theta = 2.000$ para $n = 40$	$\theta = 2.800$ para $n = 80$	$\theta = 3.600$ para $n = 120$	$\theta = 4.000$ para $n = 160$	$\theta = 4.500$ para $n = 200$
$F_s = 0,25$ para $s_{j,i,k} = 25\%$	2.500	3.500	4.500	5.000	5.625
$F_s = 0,50$ para $s_{j,i,k} = 50\%$	3.000	4.200	5.400	6.000	6.750
$F_s = 1$ para $s_{j,i,k} = 100\%$	4.000	5.600	7.200	8.000	9.000
$F_s = 1,25$ para $s_{j,i,k} = 125\%$	4.500	6.300	8.100	9.000	10.125

Fonte: Autor, 2019.

O parâmetro β recebe os valores [2,3], [2,3, 2,5], [2,3, 2,5, 3] e [2,3, 2,5 3, 3,5 e 4,5] para $l = 1$, $l = 2$, $l = 3$ e $l = 5$, respectivamente, e $t = n \times s_{j,i,k}/2 + n \times 50$. O restante dos dados dos problemas foram gerados conforme configurações da Tabela 10. Para cada configuração de n , m , l , e $s_{j,i,k}$ foram gerados 5 problemas diferentes, resultando em $5 \times 4 \times 4 \times 4 \times 5 = 1.600$ problemas por cenário de MP. Portanto, 3.200 problemas são utilizados para calibração das metaheurísticas propostas.

Todos os métodos heurísticos (construtivos e metaheurísticas) foram implementados em linguagem Delphi, por meio do *software* Delphi versão 7.0. Todos os experimentos foram realizados em um processador i5-2500k, 3.30 GHz e 4 GB de memória RAM. Os modelos matemáticos foram implementados em linguagem Python e resolvidos pela biblioteca IBM CPLEX versão 12.8. Foi estabelecido um tempo limite de 3.600 segundos para solução de cada problema. Em relação ao limite de instâncias do problema, o CPLEX foi executado até que não fosse mais encontrada solução no limite de tempo para uma dada classe de instância.

O desvio relativo (DR) foi utilizado para medir a qualidade de solução das heurísticas. O desvio relativo da solução de uma heurística a para um problema b ($DR_{a,b}$) mede o quanto a solução da heurística a ($Sol_{a,b}$) está distante da melhor solução encontrada para o problema b ($Sol_{best,b}$) e é dada por

$$DR_{a,b} = \frac{Sol_{a,b} - Sol_{best,b}}{Sol_{best,b}} \quad (48)$$

Caso $DR_{a,b} = 0$, pode-se dizer que o método a apresenta a melhor solução para o problema b . Neste sentido, quanto menor o $DR_{a,b}$, mais próxima a $Sol_{a,b}$ está de $Sol_{best,b}$, indicando uma qualidade de solução melhor do que uma outra $Sol_{a,b}$ mais distante de $Sol_{best,b}$.

Adicionalmente, foram utilizados o gráfico de médias e o gráfico de linhas para avaliar a variação ocorrida nas médias do DRM das heurísticas sob os parâmetros determinantes para geração das instâncias do problema: nível de manutenção (l), tempo de *setup* ($S_{i,j,k}$); número de máquinas (m) e número de tarefas (n). O intuito destas análises é avaliar a influência destes fatores em relação ao desempenho dos métodos. Utilizou-se o software IBM® SPSS® Statistics Versão 22 e o *Microsoft Excel* para tabulação de dados e realização de todas as análises estatísticas.

Para medir a eficiência das heurísticas, utilizou-se o tempo computacional médio dos métodos em relação ao número de tarefas. Uma análise do desempenho das heurísticas que avalia o *trade-off* entre tempo computacional e o desvio relativo médio também foi realizada

para as heurísticas construtivas, no intuito de fornecer os melhores métodos avaliados para os cenários avaliados.

4 MÉTODOS DE SOLUÇÃO PARA O PROBLEMA DE MINIMIZAÇÃO DA DURAÇÃO TOTAL DA PROGRAMAÇÃO SUJEITA AOS CUSTOS DE MANUTENÇÃO PREVENTIVA

Neste capítulo são propostas heurísticas construtivas e metaheurísticas para o problema $F_m|block, NM_j^t, s_{j,k}|(C_{max}|total_cost = C)$. Levou-se em consideração duas vias de solução: Método exato com aplicação do modelo de programação linear inteira mista (MILP) e proposição de heurísticas e metaheurísticas para o problema. Por se tratar de uma função-objetivo hierárquica, é necessário primeiro que a medida que se pretende adotar como constante seja encontrada, neste caso o custo total de manutenção preventiva (*total_cost*). Neste sentido, as heurísticas construtivas para os problemas $F_m|block|C_{max}$ e $F_m|block, s_{j,k}|C_{max}$ identificadas durante a revisão de literatura foram adaptadas e novas versões alternativas foram propostas, com o objetivo de minimizar prioritariamente o *total_cost*. Já para a minimização de C_{max} , foram propostas a adaptação de metaheurísticas desenvolvidas para os problemas anteriormente citados. Neste sentido, as heurísticas construtivas com os melhores desempenhos para cada cenário de MP avaliado foram utilizadas como *input* para as metaheurísticas.

4.1 HEURÍSTICAS CONSTRUTIVAS PARA MINIMIZAÇÃO DO CUSTO TOTAL DE MANUTENÇÃO PREVENTIVA

O uso das heurísticas construtivas tem por objetivo minimizar o custo total de manutenção preventiva (MP), *total_cost*, dadas as particularidades do problema. As seguintes heurísticas foram adaptadas: PF, PFE, MM, MM-B, MME, MME-B, PF-NEH(x), wPF-NEH(x), PW-NEH(x), PFT-NEH(x), PFT-RZ(x), PF-NEH(λ), WPT-NEH e mNEH. Uma versão alternativa para cada uma das 8 últimas heurísticas citadas foi proposta aplicando uma heurística gulosa baseada no funcionamento *Nearest Neighbor* (NN) como forma de ordenação inicial das tarefas. Em linhas gerais, NN, descrito na Figura 8, escolhe a próxima tarefa que gera o menor tempo de *setup* total na ordenação inicial.

```

 $k \leftarrow 1$ 
 $bestst \leftarrow \sum_{j=1}^m s_{[j],[1],[1]}$ 
Para  $i \leftarrow 2$  até  $n$  Faça
     $sum \leftarrow \sum_{j=1}^m s_{[j],[i],[i]}$ 
    Se  $sum < bestst$  Então
         $bestst \leftarrow sum$ 
         $k \leftarrow 2$ 
    FimSe
FimPara
 $J \leftarrow J - \{J_k\}$ 
 $\sigma \leftarrow \sigma \cup \{J_k\}$ 
Para  $i \leftarrow 2$  até  $n$  Faça
     $p \leftarrow \sigma_{[i-1]}, k \leftarrow J_{[i]}, t \leftarrow J_{[i]}$ 
     $bestst \leftarrow \sum_{j=1}^m s_{[j],p,k}$ 

    Para  $h \leftarrow 1$  até  $(n - i)$  Faça
         $k \leftarrow J_h$ 
         $sum \leftarrow \sum_{j=1}^m s_{[j],p,k}$ 
        Se  $sum < bestst$  Então
             $bestst \leftarrow sum$ 
             $k \leftarrow 2$ 
             $t \leftarrow J_{[h]}$ 
        FimSe
    FimPara
     $J \leftarrow J - \{J_t\}$ 
     $\sigma \leftarrow \sigma \cup \{J_t\}$ 
Fim

```

Figura 8 - Heurística do vizinho mais próximo para tempos de *setup*

Fonte: Autor, 2019.

Outro mecanismo comum para a maioria das heurísticas propostas é a fase de construção da solução por meio de inserção NEH, proposta por Nawaz *et al.* (1983). O pseudocódigo da heurística NEH, intitulado como *NEH_Insertion_Scheme*, pode ser visto na Figura 9.

```

// NEH_Insertion_Scheme
 $\pi \leftarrow$  Sequência de tarefas e  $\pi', \pi'' \leftarrow \emptyset$ 
//geração da lista inicial
Ordene as tarefas em  $\pi$  em ordem não-crescente do tempo total de processamento
//mecanismo de inserção NEH
Selecione as duas primeiras tarefas de  $\pi$  e insira na sequência parcial  $\pi''$ . Escolha para a próxima iteração, a
sequência parcial que minimiza a função-objetivo
Para  $i \leftarrow 3$  até  $n$  faça
    Remova a tarefa  $\pi(i)$  de  $\pi'$  e insira em todas as posições possíveis da sequência parcial  $\pi''$ . Dentre as  $i$ 
    sequências parciais geradas, escolha a que minimiza a função-objetivo e adote-a como  $\pi''$ .
FimPara
Retorne  $\pi''$ 

```

Figura 9 - Pseudocódigo da heurística NEH

Fonte: Adaptado de Nawaz *et al.* (1983).

Uma versão alternativa do mecanismo de NEH, adotado por uma parte das heurísticas construtivas, inicia o procedimento de inserção a partir de uma determinada posição λ até a última tarefa. O pseudocódigo deste procedimento pode ser visto na Figura 10. A utilização deste mecanismo justifica-se pelo fato de que *NEH_Insertion_Scheme* (Figura 9) pode piorar a qualidade de solução se aplicado a todos os elementos da lista inicial (PAN; WANG, 2012). O valor de λ é definido por meio de experimentações computacionais.

```
// NEH( $\lambda$ )
 $\pi \leftarrow$  Sequência de tarefas e  $\pi', \pi'' \leftarrow \emptyset$ 
//geração da lista inicial
 $\pi' \leftarrow$  Heurística construtivo( $\pi$ )
Para  $i \leftarrow \lambda$  até  $n$  Faça
    Remova a tarefa  $\pi(i)$  de  $\pi'$  e insira em todas as posições possíveis da sequência parcial  $\pi''$ . Dentre as  $i$ 
    sequências parciais geradas, escolha a que minimiza a função-objetivo e adote-a como  $\pi''$ .
FimPara
Retorne  $\pi''$ 
Fim // NEH( $\lambda$ )
```

Figura 10 - Pseudocódigo do mecanismo de inserção NEH(λ)_Insertion_Scheme

Fonte: Adaptado de Nawaz *et al.* (1983).

Ao longo deste capítulo, as etapas das heurísticas construtivas que utilizam o mecanismo de inserção de NEH, NEH(λ) e a heurística NN serão identificados como *NEH_Insertion_Scheme*, *NEH(λ)_Insertion_Scheme* e *NN_{setup}*, respectivamente. No total, 22 heurísticas construtivas foram propostas para a minimização do *total_cost*. O método com o melhor desempenho é então utilizado como gerador de solução(ões) inicial(is) para as metaheurísticas propostas para o problema. A seguir, descreve-se a adaptação das heurísticas para o problema proposto.

4.1.1 Profile Fitting (PF)/PFE/PF-NEH(x)/PF-NEH(x)st

A heurística PF constrói a solução testando todas as tarefas não programadas na última posição da sequência parcial. Para avaliar a qualidade das soluções, os autores adotaram a minimização da soma dos tempos de bloqueio e de ociosidade quando uma tarefa candidata J_c de J é inserida na posição i da sequência parcial:

$$\delta_{i,c} = \sum_{j=1}^m (D_{c[i]j} - D_{[i-1]j} - p_{c,j}) \quad (49)$$

Onde:

$D_{c[i]k}$ – Instante de partida da tarefa J_c de J que ocupa a i -ésima posição de S_i ;

$D_{[i-1]k}$ – Instante de partida da tarefa que ocupa a $(i - 1)$ -ésima posição de S_i ;

p_{ck} – Tempo de processamento da tarefa J_c .

Mccormick *et al.* (1989) não estabeleceram qual a primeira tarefa a ser inserida na sequência. Neste caso, posteriores publicações como a de Ronconi (2004), Companys et al. (2010), Ribas et al. (2011) e Pan e Wang (2012), determinam a primeira tarefa pelo menor tempo total de processamento. Na versão adaptada do algoritmo, a primeira tarefa selecionada é a que apresenta a menor data de conclusão na última máquina. Da segunda tarefa até a última, considera-se a minimização do custo total de MP (*total_cost*). Em caso de sequências parciais com custos iguais, a medida dada pela Eq. (49) é utilizada como critério de desempate. Uma adaptação é feita adicionando-se o tempo de *setup* e a soma das MP's em todas as máquinas ao adicionar a tarefa c na ultima posição da sequência parcial:

$$\delta_{i,c} = \sum_{j=1}^m (D_{c,j} - D_{[i-1]j} - p_{c,j} - s_{j,[i-1],c} - \sum_{t=1}^l MA_{[i-1],c,j}^t) \quad (50)$$

Posteriormente, Ronconi (2004) adicionou o mecanismo de inserção de NEH à solução fornecida por PF, criando a heurística PFE. Os pseudocódigos de PF e PFE podem ser vistos na Figura 11.

```
//PF/PFE
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$  e  $\pi'' \leftarrow \emptyset$ 
//PF
Remova a tarefa  $\pi(k)$  da sequência  $\pi$  que gera o menor  $C_{max}$  e insira em  $\pi'(1)$ 
Para  $i \leftarrow 2$  até  $n$  Faça
  Remova a tarefa  $\pi(k)$  de  $\pi$  que minimiza o total_cost. Em caso de empate, escolha a tarefa  $\pi(k)$  que minimiza  $\delta_{i,c} = \sum_{j=1}^m (D_{c,j} - D_{[i-1]j} - p_{c,j} - s_{j,[i-1],c} - \sum_{t=1}^l MA_{[i-1],c,j}^t)$  e insira em  $\pi'(i)$ 
FimPara
//PFE
Aplice NEH_Insertion_Scheme( $\pi'$ ,  $\pi''$ )
Retorne  $\pi''$ 
Fim // PF/PFE
```

Figura 11 - Pseudocódigo da heurística PF/PFE

Fonte: Adaptado de Ronconi (2004).

Durante a fase de aplicação de *NEH_Insertion_Scheme*, utiliza-se como medida de seleção da melhor sequência parcial aquela que minimiza o custo total de MP e, de forma secundária, em caso de empates, a sequência parcial que apresenta o menor C_{max} é adotada para a próxima iteração.

4.1.2 MinMax (MM)/MME, MM-B/MME-B

Ronconi (2004) propõe a heurística MinMax (MM), que emprega a minimização da diferença entre os tempos de processamento das tarefas para avaliar a qualidade de solução das sequências parciais:

$$\gamma_{[i-1],c} = \alpha \sum_{l=1}^{m-1} |p_{c,l} - p_{[i-1],l+1}| + (1 - \alpha) \sum_{j=1}^m p_{c,j} \quad (51)$$

Onde $0 \leq \alpha \leq 1$

A estratégia da heurística é tentar aumentar a probabilidade de incluir tarefas com os menores tempos de processamento no caminho crítico, posicionando-os no início da sequência. A tarefa com menor tempo total de processamento é escolhida como a primeira da sequência. A tarefa com o menor tempo de processamento na última máquina, excluindo-se a primeira tarefa já incluída na sequência parcial, é adicionada na última posição.

Na versão adaptada ao problema proposto, considerou-se o tempo de *setup* da tarefa e também a possível manutenção preventiva do tipo t a ser realizada na máquina j antes do processamento da tarefa candidata c . Neste sentido, a nova medida de avaliação para desempate das tarefas candidatas ($\gamma_{[i-1],c}$) é descrita como

$$\gamma_{[i-1],c} = \alpha \sum_{l=1}^{m-1} |p_{c,l} + s_{l,[i-1],c} + \sum_{t=1}^v MA_{[i-1],c,l+1}^t - p_{[i-1],l+1} - \sum_{t=1}^v MA_{[i-2],[i-1],l+1}^t - s_{l,[i-2],[i-1]}| + (1 - \alpha) \sum_{j=1}^m (p_{c,j} + \sum_{t=1}^l MA_{[i-1],c,l}^t + s_{l,[i-1],c}) \quad (52)$$

Em ambas as heurísticas, a primeira tarefa da sequência é definida pela menor data de término, assim como em PF. Assim como em PFE, a autora propôs a adição do mecanismo de

inserção *NEH_Insertion_Scheme* à solução fornecida por MM, propondo assim MME. O pseudocódigo da heurística MM/MME pode ser visto na Figura 12.

```

//MME
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$  e  $\pi'' \leftarrow \emptyset$ 
Remova a tarefa  $\pi(k)$  da sequência  $\pi$  que gera o menor  $C_{max}$  e insira em  $\pi'(1)$ 
Para  $i \leftarrow 2$  até  $n$  Faça
  Remova a tarefa  $\pi(k)$  de  $\pi$  que minimiza o custo total de MP. Em caso de empate, escolha a tarefa  $\pi(k)$ 
  que minimiza  $\gamma_{[i-1],c} = \alpha \sum_{l=1}^{m-1} |p_{c,l} + s_{l,[i-1],c} + \sum_{t=1}^v MA_{[i-1],c,l+1}^t - p_{[i-1],l+1} - \sum_{t=1}^v MA_{[i-2],[i-1],l+1}^t| +$ 
   $(1 - \alpha) \sum_{j=1}^m (p_{c,j} + \sum_{t=1}^v MA_{[i-1],c,l}^t + s_{l,[i-1],c})$  e insira em  $\pi'(i)$ 
FimPara
Aplique o NEH_Insertion_Scheme( $\pi'$ ,  $\pi''$ )
Retorne  $\pi''$ 
Fim //MME

```

Figura 12 - Pseudocódigo da heurística MM/MME

Fonte: Adaptado de Ronconi (2004).

Posteriormente, Han et al. (2012) propõem uma versão modificada de MM, denominada MM-B, onde a função de avaliação da próxima tarefa a ser adicionada à sequência parcial π' não considera o módulo do valor do primeiro termo. Neste sentido, o seguinte critério de desempate é estabelecido para avaliar a próxima tarefa a ser inserida na sequência parcial:

$$\gamma_{[i-1],c} = \alpha \sum_{l=1}^{m-1} \max \left(\sum_{t=1}^v MA_{[i-1],c,l+1}^t + s_{l,[i-1],c} + p_{c,l} - \sum_{t=1}^v MA_{[i-2],[i-1],l+1}^t - s_{l,[i-2],[i-1]} - p_{[i-1],l+1}, 0 \right) + (1 - \alpha) \sum_{j=1}^m \left(\sum_{t=1}^v MA_{[i-1],c,l}^t + s_{l,[i-1],c} + p_{c,j} \right) \quad (53)$$

Uma versão de MM-B com o mecanismo de inserção NEH, nomeada MME-B, também é proposta. Devido a similaridade, o pseudocódigo apresentado por MM/MME pode ser aplicado a MM-B/MME-B, modificando apenas a função de avaliação. Pelo estudo experimental apresentado pelos autores, $\alpha = 0,6$. Uma nova experimentação foi conduzida, de forma a constatar o melhor valor de MM e MM-B.

4.1.3 wPF

Pan e Wang (2012) propõem em wPF, variação da heurística PF, a avaliação da sequência por meio da soma ponderada dos tempos de bloqueio e de ociosidade, dado por

$$\delta_{i,k} = \sum_{j=1}^m w_j (D_{c[i]j} - D_{[i-1]j} - p_{c,j}) \quad (54)$$

Esse método diferencia o efeito das máquinas e tarefas em diferentes posições usando um peso de ponderação w_j associado a cada máquina j :

$$w_j = \frac{m}{\left(j + \frac{i(m-j)}{(n-2)}\right)}, \quad j = 1, 2, \dots, m \quad (55)$$

Na versão adaptada, a Equação (49) foi incorporada a Equação (54), mantendo o índice w_j inalterado:

$$\delta_{i,k} = \sum_{j=1}^m w_j (D_{c,j} - D_{[i-1]j} - p_{c,j} - s_{j,[i-1],c} - \sum_{t=1}^l MA_{[i-1],c,1}^t) \quad (56)$$

A primeira tarefa da sequência parcial é escolhida de acordo com a menor data de conclusão. O pseudocódigo de wPF pode ser visto na Figura 13.

```
//wPF
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$  e  $\pi'' \leftarrow \emptyset$ 
Remova a tarefa  $\pi(k)$  da sequência  $\pi$  que gera o menor  $C_{max}$  e insira em  $\pi'(1)$ 
Para  $i \leftarrow 2$  até  $(n-1)$  Faça
  Calcule o valor de  $w_j = m / (j + k(m-j)/(n-2))$ , onde  $j = 1, \dots, m$ .
  Calcule o instante de partida  $D_{[i]j}$ ,  $j = 1, \dots, m$ , ao adicionar a tarefa  $\pi(k)$  em  $\pi'(i)$ .
  Selecione a tarefa  $\pi(k)$  que minimiza o total_cost. Em caso de empate, escolha a tarefa que minimiza a
  função  $\delta_{i,k} = \sum_{j=1}^m w_j (D_{c,j} - D_{[i-1]j} - p_{c,j} - s_{j,[i-1],c} - \sum_{t=1}^l MA_{[i-1],c,1}^t)$  e insira em  $\pi'(i)$ 
FimPara
Insira a última tarefa restante de  $\pi$  na última posição de  $\pi'$ 
Retorne  $\pi'$ 
End // wPF
```

Figura 13 - Pseudocódigo da heurística wPF

Fonte: Adaptado de Pan e Wang (2012).

4.1.4 PW

Para a heurística PW, Pan e Wang (2012) propõem a adição de uma tarefa artificial v para representar o peso das tarefas ainda não programadas do conjunto J . Baseada em Liu e

Reeves (2001), a tarefa artificial v é anexada na posição $i + 1$ da sequencia parcial, após a tarefa q de J ser inserida na posição i . Seu tempo de processamento é representado pela média dos tempos de processamento das k tarefas restantes $(J - q)$ de J :

$$p_{v,j} = \sum_{\substack{k \in J \\ k \neq q}} p_{q,j} / (n - i - 1) \quad j = 1, 2, \dots, m \quad (57)$$

Com as datas de partida da tarefa v ($D_{[i+1],j}$, $j = 1, 2, \dots, m$), os autores calculam a soma dos tempos de ociosidade e bloqueio, $\chi_{i,k}$ causado pela tarefa v :

$$\chi_{i,k} = \sum_{j=1}^m w_j (D_{[i+1],j} - D_{[i],j} - p_{v,j}) \quad (58)$$

Os autores usam a medida $\chi_{i,k}$ para avaliar o efeito da tarefa v nas tarefas que ainda não foram programadas. Além disso, os autores também avaliam o efeito da tarefa candidata c , usando a mesma medida $\delta_{i,k}$ dada por wPF. Combinando as duas medidas $\delta_{i,k}$ e $\chi_{i,k}$, Pan e Wang (2012) chegam ao índice $f_{i,k}$, dado por

$$f_{i,k} = \sum_{j=1}^m (n - k - 2) \delta_{i,k} + \chi_{i,k} \quad (59)$$

onde $\delta_{0,1} = \sum_{j=1}^m w_j (D_{[1],j} - p_{i,j})$.

A medida que o número de tarefas diminui, $\chi_{i,k}$ reflete melhor o efeito das tarefas a serem posteriormente programadas, logo, a contribuição de $\delta_{i,k}$ diminuindo a prioridade ao valor de $\chi_{i,k}$. A tarefa selecionada é a que apresenta o mínimo $f_{i,k}$.

Para adaptar a heurística PW ao problema, é necessário que a tarefa artificial, além do seu tempo de processamento, também possua seu respectivo tempo de *setup* em cada máquina. Com esses valores em mãos, também é possível estimar as operações de MP necessárias para processar a tarefa artificial v . Por se tratar de um valor que depende da sequência de tarefas, o tempo de *setup* artificial proposto é composto pela soma dos tempos de *setup* entre a última tarefa programada na sequencia parcial e as tarefas ainda não programadas:

$$st_{artj} = (\sum_{k=1}^{n-i} s_{j,[i-1],k}) / (n - i) \quad j = 1, \dots, m \quad (60)$$

Uma experimentação computacional prévia revelou que a adição de uma “MP artificial” à função de avaliação conduziu a soluções com qualidade pior que se considerado apenas os tempos de processamento e de *setup* artificiais. Portanto, $\chi_{i,k}$ é calculado como:

$$\chi_{i,k} = \sum_{j=1}^m w_j (D_{v,j} - D_{c,j} - p_{v,j} - st_{artj}) \quad (61)$$

Por fim, $f_{i,k}$ pode ser calculado por meio da Eq. (59). O processo de seleção da primeira tarefa da sequência segue o proposto para PF, MM e wPF. O pseudocódigo da heurística PW pode ser visto na Figura 14.

```
//PW
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$  e  $\pi'' \leftarrow \emptyset$ 
Remova a tarefa  $\pi(k)$  que apresenta o menor valor  $f_{i,0}$  de  $\pi(k)$  e insira em  $\pi'(1)$ . Caso haja tarefas com o mesmo valor de  $f_{i,0}$  selecione aquela que minimiza  $\chi_{i,0}$ 
Para  $i \leftarrow 2$  até  $(n-1)$  Faça
  Calcule o instante de partida  $D_{[i],j}$ ,  $j = 1, \dots, m$ , ao adicionar a tarefa  $\pi(k)$  em  $\pi'(i)$ .
  Para cada tarefa  $\pi(k) \in \pi$ , calcule o tempo de processamento  $p_{v,j}$  da tarefa artificial, seu respectivo tempo de setup  $st_{artj}$  para cada máquina e o instante de partida  $D_{[i+1],j}$  da tarefa artificial em todas as máquinas.
  Calcule o valor de  $f_{i,k}$ .
  Selecione a tarefa que minimiza o total_cost. Em caso de empates escolha a tarefa que minimiza  $f_{i,k}$ . Utilize  $\chi_{i,k}$  em caso de empates, remova a tarefa  $\pi(k)$  de  $\pi$  e insira em  $\pi'(i)$ 
FimPara
Insira a última tarefa restante de  $\pi$  na última posição de  $\pi'$ 
Retorne  $\pi'$ 
End // PW
```

Figura 14 - Pseudocódigo da heurística PW

Fonte: Adaptado de Pan e Wang (2012).

4.1.5 PF-NEH(x)/PF-NEH(x)st, wPF-NEH(x)/wPF-NEH(x)st e PW-NEH(x)/PW-NEH(x)st

Pan e Wang (2012) propõem em PF-NEH(x), wPF-NEH(x) e PW-NEH(x), a geração da sequência inicial por meio da heurística PF, wPF e PW, respectivamente. Em PF-NEH, wPF-NEH e PW-NEH, a construção da solução inicia-se a partir da tarefa que ocupa a posição $(n - \lambda + 1)$ da sequência R_i , onde λ é parametrizável. Os autores utilizam o mecanismo de inserção NEH(λ)_Insertion_Scheme para as λ tarefas. Em PF-NEH(x), wPF-NEH(x) e PW-

NEH(x), os autores estabelecem uma sequência inicial em que as tarefas são ordenadas em ordem não-decrescente dos tempos totais de processamento. O parâmetro x define o número de vezes que PF-NEH, wPF-NEH e PW-NEH é executado. Para um dado x , a heurística escolhe a x -ésima tarefa da lista inicial para ser a primeira tarefa da sequência a ser construída por PF, wPF ou PW. A melhor sequência entre as x sequências geradas, ou seja, que obteve o menor *total_cost* é dada como solução do problema. Uma versão alternativa de cada heurística, nomeadas PF-NEH(x)st, wPF-NEH(x)st e PW-NEH(x)st, que emprega NN_{setup} para geração da lista inicial, é proposta. O pseudocódigo das heurísticas PF-NEH(x)\PF-NEH(x)st, wPF-NEH(x)\wPF-NEH(x)st e PW-NEH(x)\PW-NEH(x)st pode ser vista na Figura 15.

```

//PF-NEH(x)/PF-NEH(x)st/wPF-NEH(x)/wPF-NEH(x)st/PW-NEH(x)/PW-NEH(x)st
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$ ,  $\pi''$  e  $\pi_{best} \leftarrow \emptyset$ 
Ordene as tarefas em  $\pi$  ordem não-crescente do tempo total de processamento/Aplique  $NN_{setup}$  em  $\pi$ 
 $bestcmax, bestcost \leftarrow$  Número muito grande
Para  $p \leftarrow 1$  até  $x$  Faça
   $\pi'(1) \leftarrow \pi(p)$ 
   $\pi' \leftarrow$  PF( $\pi$ )/wPF( $\pi$ )/PW( $\pi$ )
  //Inserção NEH( $\lambda$ )
  Aplique NEH( $\lambda$ )_Insertion_Scheme( $\pi'$ ,  $\pi''$ )
  Se  $bestcost < total\_cost(\pi'')$ 
     $bestcost \leftarrow total\_cost(\pi'')$ ,  $bestcmax \leftarrow C_{max}(\pi'')$ 
     $\pi_{best} \leftarrow \pi''$ 
  Senão
    Se  $bestcost = total\_cost(\pi'')$  and  $bestcmax > C_{max}(\pi'')$  Então
       $bestcost \leftarrow total\_cost(\pi'')$ ,  $bestcmax \leftarrow C_{max}(\pi'')$ 
       $\pi_{best} \leftarrow \pi''$ 
  FimSe
FimSenão
FimPara
Retorne  $\pi_{best}$ 
Fim // PF-NEH(x)/PF-NEH(x)st/wPF-NEH(x)/wPF-NEH(x)st/PW-NEH(x)/PW-NEH(x)st

```

Figura 15 - Pseudocódigo das heurísticas PF-NEH(x), wPF-NEH(x) e PW-NEH(x)

Fonte: Autor, 2019.

Os autores conduziram uma experimentação computacional e chegaram ao valor $\lambda = 25$ para PF-NEH(x) e $\lambda = 20$ para wPF-NEH(x) e PW-NEH(x) como os melhores. O parâmetro x foi estabelecido como $x = 1, 2$ e 5 . Logo, se $x = 5$, as heurísticas são rodadas cinco vezes, gerando cinco sequências de acordo com o algoritmo apresentado na Figura 15.

4.1.6 Modified NEH (mNEH)/mNEHst

Proposta por Wang *et al.* (2012), a heurística Modified NEH leva em conta uma nova regra de prioridade para geração da lista inicial. Por experimentações computacionais, os autores constataram que desvios padrões mais altos do tempo de processamento tendem a gerar uma alta probabilidade de bloqueio. Os autores então criaram uma nova regra de prioridade que inclui a média dos tempos de processamento e seus respectivos desvios padrões. A seguinte medida é utilizada para gerar o índice da tarefa:

$$P_i = \alpha AVG_i + (1 - \alpha) STD_i$$

$$AVG_i = \left(\frac{1}{m} \sum_{j=1}^m p_{i,j} \right) \text{ e } STD_i = \sqrt{\frac{1}{(m-1)} \sum_{j=1}^m (p_{i,j} - AVG_i)^2}$$

Onde o parâmetro α é estabelecido entre a faixa [0,1] e que determina a prioridade da medida no índice. A lista inicial é estabelecida de acordo com a ordem não decrescente de P_i . Uma versão alternativa, denominada, mNEH_{st}, que leva em consideração as médias e desvios relativos dos tempos de *setup* das tarefas é proposta e pode ser aplicada ao algoritmo NN_{setup} (Figura 8). As seguintes medidas são utilizadas para substituir o tempo de *setup* $s_{j,i,k}$:

$$P_i = \alpha AVG_i + (1 - \alpha) STD_i \quad (62)$$

$$AVG_i = \left(\frac{1}{m} \sum_{j=1}^m s_{j,i,k} \right) \text{ e } STD_i = \sqrt{\frac{1}{(m-1)} \sum_{j=1}^m (s_{j,i,k} - AVG_i)^2}$$

O pseudocódigo de NEH (Figura 9) pode ser aplicado para ambos mNEH e mNEH_{st}, modificando-se apenas a geração da lista inicial e a função-objetivo adotada que consiste prioritariamente na minimização do *total_cost*, e em caso de desempate, a sequência parcial que minimiza C_{max} .

4.1.7 WPT-NEH/WPT-NEH_{st}

A heurística WPT-NEH foi proposta por Wang et al. (2010) para o problema $F_m|block|\sum_i C_i$ e leva em consideração uma modificação da lista inicial, ordenando as tarefas em ordem não-decrescente do tempo total de processamento. Uma versão alternativa desta heurística é proposta, denominada WPT-NEH_{st}, na qual o algoritmo NN_{setup} é utilizado para gerar a lista inicial. Deste modo, o pseudocódigo de NEH (Figura 9) pode ser aplicado para WPT-NEH e WPT-NEH_{st}, modificando apenas a geração da lista inicial.

4.1.8 PFT-NEH(x)/PFT-NEH(x)_{st} e PFT-RZ(x)/PFT-RZ(x)_{st}

A heurística PFT-NEH(x), proposta por Tasgetiren et al. (2017), possui funcionamento análogo a das heurísticas propostas por Pan e Wang (2012). Em PFT, as tarefas são inicialmente ordenadas de acordo com o tempo de espera pelo processamento da primeira tarefa em todas as máquinas (*front delay*), definido como:

$$P_i = \frac{2}{(m-1)} \left(\sum_{j=1}^m (m-j)p_{i,j} \right) + \sum_{j=1}^m (m-j)p_{i,j} \quad (63)$$

A lista inicial é ordenada de acordo com a ordem não-decrescente de P_i . O primeiro elemento da lista inicial é inserido na sequência parcial e o restante das tarefas é selecionado de forma similar a de PF. A função de avaliação para desempate de sequência com custos totais de MP iguais leva em conta o tempo de ociosidade e bloqueio total entre máquinas, ou seja, $\delta_{i,c} = \sum_{j=1}^m (D_{c,j} - D_{[i-1],j} - p_{c,j} - s_{j,[i-1],c} - \sum_{t=1}^l MA'_{[i-1],c,j})$. A função de avaliação é definida por:

$$\mu_{i,c} = (n-i-2) \times \delta_{i,c} + C_{i,m} \quad (64)$$

Em PFT-NEH, o mecanismo de inserção $NEH(\lambda)_{Insertion_Scheme}$ é aplicado às λ últimas tarefas da sequência dada por PFT. Os autores estabeleceram que $\lambda = 20$ tarefas. Por fim, PFT-NEH(x) gera x sequências a partir da lista inicial, de modo análogo as heurísticas propostas por Pan e Wang (2012). Uma versão alternativa, denominada PFT-NEH(x)_{st} é

proposta, empregando NN_{setup} como gerador da lista inicial de tarefas. O pseudocódigo da heurística PFT-NEH(x) pode ser visto na Figura 16.

```

//PFT/PFT-NEH(x)/PFT-NEH(x)st
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$ ,  $\pi''$  e  $\pi_{best} \leftarrow \emptyset$ 
Ordene as tarefas em  $\pi$  ordem não-decrescente de  $P_i = \frac{2}{(m-1)} \left( \sum_{j=1}^m (m-j)p_{i,j} \right) + \sum_{j=1}^m (m-j)p_{i,j}$  / Aplique  $NN_{setup}$ 
em  $\pi$  utilizando como medida de avaliação  $P_i = \frac{2}{(m-1)} \left( \sum_{j=1}^m (m-j)s_{j,i,k} \right) + \sum_{j=1}^m (m-j)s_{j,i,k}$ 
 $bestcmax, bestcost \leftarrow$  Número muito grande
Para  $p \leftarrow 1$  até  $x$  Faça
   $\pi'(1) \leftarrow \pi(p)$ 
  Para  $i \leftarrow 2$  até  $n$  Faça
    Remova a tarefa  $\pi(k)$  de  $\pi$  que minimiza o custo total de MP. Em caso de empate, escolha a tarefa  $\pi(k)$ 
    que minimiza  $\mu_{i,c} = (n-i-2) \times \delta_{i,c} + C_{i,m}$  e insira em  $\pi'(i)$ 
  FimPara
  //Inserção NEH( $\lambda$ )
  Aplique NEH( $\lambda$ )_Insertion_Scheme( $\pi'$ ,  $\pi''$ )
  Se  $bestcost < MP_{cost}(\pi')$  Então
     $bestcost \leftarrow MP_{cost}(\pi')$ ,  $bestcmax \leftarrow C_{max}(\pi')$ 
     $\pi_{best} \leftarrow \pi'$ 
  Senão
    Se  $bestcost = MP_{cost}(\pi')$  and  $bestcmax > C_{max}(\pi')$  Então
       $bestcost \leftarrow MP_{cost}(\pi')$ ,  $bestcmax \leftarrow C_{max}(\pi')$ 
       $\pi_{best} \leftarrow \pi'$ 
    EndSe
  EndSenão
FimPara
Retorne  $\pi_{best}$ 
End //PFT/PFT-NEH(x)/PFT-NEH(x)st

```

Figura 16 - Pseudocódigo da heurística PFT-NEH(x)

Fonte: Autor, 2019.

Já em PFT-RZ(x), proposta por Shao *et al.* (2018a), o mecanismo de inserção de RZ de Rajendran e Ziegler (1997) é aplicado à solução dada por PFT. O pseudocódigo de PFT-RZ(x) pode ser visto na Figura 17.

```

//PFT-RZ(x)/PFT-RZ(x)st
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$ ,  $\pi''$  e  $\pi_{best} \leftarrow \emptyset$ 
Ordene as tarefas em  $\pi$  ordem não-decrescente de  $P_i = \frac{2}{(m-1)} \left( \sum_{j=1}^m (m-j)p_{i,j} \right) + \sum_{j=1}^m (m-j)p_{i,j}$  / Aplique  $NN_{setup}$ 
em  $\pi$ 
 $bestcmax, bestcost \leftarrow$  Número muito grande
Para  $p \leftarrow 1$  até  $x$  Faça
   $\pi'(1) \leftarrow \pi(p)$ 
   $\pi' \leftarrow PFT(\pi, \pi')$ 
   $\pi \leftarrow \pi'$ 
  //Inserção RZ
  Para  $i \leftarrow 1$  até  $n$  Faça
    Remova a tarefa  $\pi'(k) = \pi(i)$  em  $\pi'$  e insira em todas as posições possíveis de  $\pi'$ . Escolha a sequência  $\pi''$ 
    que minimiza o custo total de MP. Caso haja empate, escolha a sequência que minimiza  $C_{max}$ .
    Se  $MPcost(\pi'') \leq MPcost(\pi')$  and  $C_{max}(\pi'') < C_{max}(\pi')$  Então
       $\pi' \leftarrow \pi''$ 
    FimSe
  FimPara
  Se  $bestcost < MP_{cost}(\pi')$  Então
     $bestcost \leftarrow MP_{cost}(\pi'), bestcmax \leftarrow C_{max}(\pi')$ 
     $\pi_{best} \leftarrow \pi'$ 
  Senão
    Se  $bestcost = MP_{cost}(\pi')$  and  $bestcmax > C_{max}(\pi')$  Então
       $bestcost \leftarrow MP_{cost}(\pi'), bestcmax \leftarrow C_{max}(\pi')$ 
       $\pi_{best} \leftarrow \pi'$ 
    FimSe
  FimSenão
FimPara
Retorne  $\pi_{best}$ 
Fim // PFT-RZ(x)/PFT-RZ(x)st

```

Figura 17 - Pseudocódigo da heurística PFT-RZ(x)/PFT-RZ(x)_{st}

Fonte: Autor, 2019.

4.1.9 PF+NEH(λ)/ PF+NEH(λ)_{st}

A heurística PF+NEH(λ) foi proposta por Pan et al. (2013) e une as estruturas de PF e NEH. Em linhas gerais, o método utiliza PF para programar as λ primeiras tarefas da sequência e as $n - \lambda$ tarefas restantes são sequenciadas conforme *NEH(λ)_Insertion_Scheme*. Uma versão alternativa, denominada PF+NEH(λ)_{st}, utiliza NN_{setup} para gerar a lista inicial na fase NEH. Na Figura 18 é mostrado o pseudocódigo de ambas heurísticas. Os autores estabeleceram $\lambda = 20$ tarefas. Em casos de instâncias com n menores que 20 tarefas, o algoritmo aplica NEH ao problema.

```

// PF+NEH( $\lambda$ )/PF+NEH( $\lambda$ )st
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$  e  $\pi'' \leftarrow \emptyset$ 
 $\pi' \leftarrow$  Aplique PF( $\pi$ ) para as primeiras  $\lambda$  tarefas da sequência  $\pi$ 
 $\pi \leftarrow$  Ordene as  $n - \lambda$  na ordem não-decrescente do tempo total de processamento das tarefas em  $\pi$  /Ordene
as  $n - \lambda$  de acordo com  $NN_{\text{setup}}$  em  $\pi$ , partindo-se da última tarefa da sequência parcial  $\pi'$ 
 $\pi' \leftarrow$  Aplique o NEH( $\lambda$ )_Insertion_Scheme( $\pi$ ,  $\pi''$ )
Retorne  $\pi'$ 
Fim // PF+NEH( $\lambda$ )/PF+NEH( $\lambda$ )st

```

Figura 18 - Pseudocódigo da heurística PF+NEH(λ)/PF+NEH(λ)st

Fonte: Adaptado de Pan et al. (2013).

4.2 METAHEURÍSTICAS

Baseado na revisão de literatura apresentada no Capítulo 2, duas recentes metaheurísticas foram adaptadas ao problema estudado: *Iterated Greedy* de Tasgetiren *et al.* (2017) e *Discrete Water Wave Optimization* de Shao et al. (2018a), provados como os mais eficazes para os problemas $F_m|block|C_{max}$ e $F_m|block, s_{j,i,k}|C_{max}$, respectivamente. Na seções posteriores os métodos serão descritos.

4.2.1 *Iterated Greedy* (IG)

O algoritmo *Iterated Greedy* foi originalmente desenvolvido por Ruiz e Stützle (2007) e sua versão básica pode ser vista na Figura 19.

```

//IG Tradicional
 $\pi_0 \leftarrow$  Sequência atual
 $\pi \leftarrow$  Busca_Local( $\pi_0$ )
Faça
   $\pi' \leftarrow$  Destruição/Construção( $\pi/d$ )
   $\pi'' \leftarrow$  Busca_Local( $\pi$ )
   $\pi \leftarrow$  Critério_de_Aceitação( $\pi$ )
Enquanto(Critério_de_Parada)
  Retorne  $\pi$ 

```

Figura 19 - Pseudocódigo do algoritmo IG

Fonte: Adaptado de Tasgetiren *et al.* (2017)

Conforme descrito na Figura 19, novas soluções são geradas da seguinte maneira: A solução atual π passa por uma fase de destruição/reconstrução, onde d tarefas são aleatoriamente removidas de π e são reinseridas nas $n-d-1$ posições de π , por meio do mecanismo de inserção $NEH(\lambda)_{Insertion_Scheme}$, até que uma nova solução π' seja encontrada. A nova solução π' passa então por uma fase de melhoria, na qual são aplicadas buscas locais, afim de buscar em sua vizinhança uma solução π'' que melhora π' . Se a nova sequência π'' é melhor que π , a solução atual é substituída, caso contrário, um critério de aceitação é aplicado, dada uma probabilidade de que π'' seja aceita. O procedimento então é repetido com a solução atual, até que um critério de parada seja atendido. De acordo com Tasgetiren *et al.* (2017), duas versões do método IG apresentaram os melhores desempenhos: IG_IJ e IG_RIS. Ambos os métodos foram adotados e adaptados para o problema $F_m|block, NM_j^t, s_{j,k}|(C_{max}|MP_{cost} = C)$ e seus pseudocódigos podem ser vistos na Figura 20.

```

//IG_IJ e IG_RIS
 $\pi_0, \pi^R, \pi_{best} \leftarrow$  Solução Inicial
 $MP_{cost} \leftarrow total\_cost(\pi_0)$ 
Enquanto (notTermination) Faça
  //construção destruição
   $\pi_1 \leftarrow$  Destruição/Construção( $\pi_0, d$ )
  //Busca local
  Se ( $r \leq jP$ ) Então Faça
     $\pi_2 \leftarrow$  RLS( $\pi_1, \pi^R$ );
  Senão
     $\pi_2 \leftarrow$  BestSwap( $\pi_1$ );
  FimSe
  //Critério de aceitação
  Se  $total\_cost(\pi_1) \leq MP_{cost}$  and  $C_{max}(\pi_2) < C_{max}(\pi_0)$  Então
     $\pi_0 \leftarrow \pi_2$ 
     $\pi_{best} \leftarrow \pi_0$ 
     $\pi^R \leftarrow \pi_{best}$ 
  Senão
    Se  $C_{max}(\pi_2) > C_{max}(\pi_{best})$  and  $total\_cost(\pi_2) \leq MP_{cost}$  Então
      Se ( $r < \exp\{-(C_{max}(\pi_2) - C_{max}(\pi_0))/T\}$ ) Então
         $\pi_0 \leftarrow \pi_2$ 
      FimSe
    FimSe
  FimSenão
FimEnquanto
Retorne  $\pi_{best}$ 
Fim // IG_IJ e IG_RIS

```

Figura 20 - Pseudocódigo de IG_IJ e IG_RIS

Fonte: Autor, 2019.

Durante a aplicação do mecanismo de inserção NEH na fase de destruição/reconstrução, o algoritmo requer uma função de avaliação para escolha da melhor sequencia parcial. Logo, não há possibilidade de utilizar o custo total ($total_cost = C$) estabelecido como restrição. Então, o seguinte critério foi estabelecido: O custo da primeira sequência parcial é adotado como melhor $total_cost$. Se uma nova sequencia parcial apresentar um $total_cost$ menor ou igual e C_{max} menor que o da atual, a sequencia parcial candidata é adotada como atual.

<pre> //Referenced Insertion Search i ← 1 π', π'' ← Sequência candidata π^R ← Melhor sequência encontrada pos ← 1, pt1 ← -1, pt2 ← -1 Enquanto(i ≤ n) Faça k ← 1 Enquanto (π'_k ≠ π^R_{pos}) k ← k + 1 FimEnquanto pos ← (pos + 1)(mod)n π'₁ ← Remova a tarefa π'_k de π' Para q ← 1 até n Faça π'₂ ← insira a tarefa π'_k na posição q de π'₁ if C_{max}(π') > C_{max}(π'₂) and MP_{cost} ≥ total_cost(π₂) Então pt1 ← k, pt2 ← q C_{max}(π') ← C_{max}(π'₂) FimPara if (pt1 > -1) Faça π'' ← insira a tarefa π'_{pt1} na posição pt2 de π''₁ i ← 1 Senão i ← i + 1 FimSe FimEnquanto Retorne π'' Fim //Referenced Insertion Search </pre>	<pre> //BestSwap π' ← Sequência candidata Para (i ← 1 até n - 1) Faça Para (k ← i + 1 até n) Faça troca_entre_pares(π', i, j) if C_{max}(π*) < C_{max}(π'') and and MP_{cost} ≥ total_cost(π*) Então pt1 ← i, pt2 ← k C_{max}(π'') ← C_{max}(π*) FimSe FimSe FimSe FimPara FimPara troca_entre_pares(π', pt1, pt2) π'' ← π' Retorne π'' Fim //BestSwap </pre>
---	---

Figura 21 - Pseudocódigo das busca locais: a) RIS e b) BestSwap

Em IG_IJ, duas estruturas de vizinhança baseadas em inserção e troca entre pares de tarefas denominadas *Referenced Insertion Search* (RIS) e *Best Swap Search* (BestSwap), respectivamente, são adotadas na fase da busca local. Os pseudocódigos de ambas buscas locais estão descritos nos quadros a) e b) da Figura 21. A decisão de escolha de qual das duas buscas locais será aplicada depende de uma probabilidade de salto jP . Na busca local RIS, uma sequencia referência π^R é utilizada para guiar o processo de busca de melhores solução na vizinhança da sequencia candidata π' . A idéia básica por trás da busca local RIS é tentar

encontrar posições melhores para tarefas em π' utilizando π^R como modelo de referência. Como exemplo de aplicação, suponha-se que $\pi^R = \{J_6, J_2, J_4, J_1, J_5, J_3\}$ e a sequência atual é dada por $\pi' = \{J_3, J_5, J_1, J_4, J_2, J_6\}$. A busca local RIS remove primeiro a tarefa J_6 da sequência candidata e a insere em todas as posições possíveis de π' . Então, na próxima iteração, J_2 é removida de π' e testada em todas as posições possíveis. O procedimento é repetido até que todas as tarefas em π^R sejam testadas e uma nova sequência π'' seja gerada. Na busca local BestSwap (Figura 21.b), a tarefa na posição i da sequência π' é trocada de posição com a tarefa na posição k , tal que $k > i$. O procedimento é repetido até que a penúltima tarefa seja testada. A melhor solução encontrada das $n \times (n - 1)/2$ sequências geradas é adotada como π'' . Em ambas as buscas locais, uma solução é avaliada somente se o custo total da solução candidata for menor ou igual a restrição do custo total MP_{cost} . Após a fase da busca local, um critério de avaliação é aplicado para verificar se a nova sequência π'' pode substituir π_0 . Os autores adotam o critério de aceitação do algoritmo *Simulated Annealing*, de Osman e Potts (1989), definido como

$$T = \frac{\sum_{i=1}^n \sum_{j=1}^m p_{i,j}}{10 \times n \times m} \times \tau P \quad (65)$$

onde T é uma Temperatura constante e τP é um parâmetro a ser ajustado

O algoritmo IG com probabilidade de salto de iteração é denotado como IG_IJ. Se o parâmetro jP assume valores menores que um número real aleatório r , a busca local BestSwap é aplicada, caso contrário, RIS é utilizada. Quando jP assume valor zero, IG aplica somente RIS na fase de busca local e será denominada como IG_RIS.

4.2.2 Discrete Water Wave Optimization (DWWO)

A metaheurística *Water Wave Optimization* (WWO) é um algoritmo populacional inspirado pelo modelo do movimento de ondas rasas, considerando a força do vento, interações de ondas não-lineares e dissipação friccional. Originalmente, WWO foi proposta para problemas de otimização contínua e sua estrutura básica pode ser vista em Zheng (2015). Conforme apresentado pelo autor supracitado, cada onda representa um elemento da população e possui uma altura h e um comprimento de onda λ . O método tenta imitar o movimento de três

tipos de movimentos de ondas incluindo propagação, quebra e refração como forma de busca de soluções do algoritmo. Na fase de propagação, as ondas com valor *fitness* alto tendem a se mover para localidades próximas para explorar áreas menores, enquanto ondas com baixo valor de *fitness* tendem a se mover para localidades remotas, explorando áreas maiores. Na fase de refração, ondas estacionárias perdem sua energia e refratam para posições aleatórias para evitar a estagnação e aumentar a diversidade da população. Na fase de quebra, uma busca intensiva é executada em torno da melhor onda para tentar melhorar a qualidade das soluções. Essas três fases são repetidas iterativamente até que um critério de parada seja atendido.

```
//DWWO
Inicialize os parâmetros  $\lambda_{min}$ ,  $\lambda_{max}$ ,  $\omega$ ,  $h_{max}$ ,  $PS$ ,  $MP_{cost}$  e a população  $\Pi = \{\pi_1, \dots, \pi_{PN}\}$ 
 $\pi_{best} \leftarrow$  Melhor sequência da população  $\Pi$ 
Repita
  Para  $k \leftarrow 1$  até PN Faça
     $\pi' \leftarrow$  Propagação com duas fases ( $\pi_k$ ) // Propagação
    Se  $MP_{cost} \geq total\_cost(\pi')$  and  $(C_{max}(\pi') < C_{max}(\pi_k))$  Então
      Se  $MP_{cost} \geq total\_cost(\pi')$  and  $(C_{max}(\pi') < C_{max}(\pi_{best}))$  Então
         $\pi'' \leftarrow VNS(\pi')$ 
         $\pi_{best} \leftarrow \pi''$ 
      EndSe
      Se  $\{\pi'\} \cap \Pi = \emptyset$  Então
        Se  $MP_{cost} \geq total\_cost(\pi')$  and  $(C_{max}(\pi') < C_{max}(\pi_{worst}))$  Então
          Substitua o pior indivíduo  $\pi_{worst}$  em  $\Pi$  por  $\pi'$ 
           $\pi_{worst}[h] \leftarrow h_{max}$ 
        EndSe
      EndSe
    EndSe
  Senão
     $\pi_k[h] \leftarrow \pi_k[h] - 1$ 
    Se  $\pi_k[h] = 0$  Então
       $\pi_k \leftarrow Path\_relinking(\pi_k, \pi_{best})$  // Refração
       $\pi_k[h] \leftarrow h_{max}$ 
    Senão
       $\pi_{worst} \leftarrow$  Encontre o pior indivíduo da população  $\Pi$ 
      Se  $\{\pi'\} \cap \Pi = \emptyset$  Então
        Se  $MP_{cost} \geq total\_cost(\pi')$  and  $(C_{max}(\pi') < C_{max}(\pi_{worst}))$  Então
          Substitua o pior indivíduo  $\pi_{worst}$  em  $\Pi$  por  $\pi'$ 
           $\pi_{worst}[h] \leftarrow h_{max}$ 
        EndSe
      EndSe
    EndSe
  EndSe
FimPara
Até(Critério_de_Parada)
Retorne  $\pi_{best}$ 
End //DWWO
```

Figura 22 - Pseudocódigo de DWWO

Fonte: Adaptado de Shao *et al.* (2018a).

Uma versão de WWO para problemas de otimização discreta (DWWO) foi proposta por Shao *et al.* (2018a) para o problema $F_m|block, s_{j,i,k}|C_{max}$ e será adaptada ao problema $F_m|block, NM_j^t, s_{j,k}|(C_{max}|total_cost = C)$. A estrutura da DWWO proposta por Shao *et al.* (2018a) pode ser vista na Figura 22. DWWO inicia a partir de uma população com PS indivíduos. Cada onda (indivíduo) na população é representado por uma sequência de tarefas. Para garantir uma população com alto nível de diversidade e qualidade, o primeiro indivíduo é fonecido por um método heurístico, enquanto os outros são gerados por meio de uma de perturbação simples no primeiro indivíduo. O parâmetro MP_{cost} corresponde ao custo deste primeiro indivíduo. O mecanismo de perturbação proposto por Nagano *et al.* (2012) consiste de dois movimentos: o primeiro consiste em selecionar uma tarefa e trocá-la de posição com sua adjacente seguinte; o outro consiste em selecionar duas tarefas aleatoriamente e trocá-las de posição. Para evitar mudanças severas, as duas tarefas estão posicionadas entre um máximo de $\max(n/50, 30)$ posições. O mecanismo é aplicado somente uma vez para cada novo indivíduo. A cada iteração, os indivíduos são ordenados de acordo com o mínimo $total_cost$. Como critério de desempate, é tida como melhor solução aquela que apresenta o menor C_{max} .

4.2.2.1 Propagação

A fase de propagação tem um papel importante na criação de novas ondas afetando de forma significativa as habilidades de busca do algoritmo. Shao *et al.* (2018a) propõe dois estágios para esta fase. No primeiro estágio, um procedimento de perturbação, que consiste em um número de d inserções aleatórias, é realizada na solução atual no intuito de gerar novas ondas. Quando d assume um valor grande, o algoritmo encontra a vizinhança das ondas em áreas mais remotas, ao passo que um valor pequeno de d explora a vizinhança próxima da área. O valor de d_i para a i -ésima onda da população é determinado por

$$d_i = \lfloor \lambda_{min} + (1 - e^{-\Delta T_i}) \times (\lambda_{max} - \lambda_{min}) \rfloor \quad (66)$$

$$\Delta T_i = \frac{f_i - f_{best} + \varepsilon}{f_{worst} - f_{best} + \varepsilon} \quad (67)$$

Onde f_i corresponde a função *fitness* da i -ésima onda, f_{best} e f_{worst} a função *fitness* do melhor e do pior indivíduo na população em uma dada iteração, respectivamente, λ_{max} e λ_{min} é o comprimento de onda máximo e mínimo, respectivamente e ε é uma constante muito pequena, usada para evitar a divisão por zero. De acordo com a Eq. (67), fica claro que quando $f_i \leftarrow f_{best}$, então $\Delta T_i \rightarrow 0$ e, conseqüentemente $d_i \leftarrow \lambda_{min}$, o que indica que ondas próximas ao melhor indivíduo tem um pequeno valor de d , resultando na criação de ondas dentro de uma vizinhança muito próxima de π_{best} . Por outro lado, quando f_i está distante de f_{best} , então $d_i \leftarrow \lambda_{max}$, o que gera indivíduos sob uma amplitude mais larga. Devido a diversidade das soluções, f_i, f_{best} e f_{worst} correspondem ao *total_cost* do i -ésimo, melhor e pior indivíduo, respectivamente. A adoção do custo como função *fitness* é justificada pelo fato de que soluções com custos piores que a restrição MP_{cost} podem apresentar C_{max} menores que indivíduos que possuem custos dentro da restrição, o que pode infactibilizar o funcionamento do algoritmo. Para evitar que o procedimento de perturbação aleatória obtenha soluções muito pobres, um número ω de novas ondas são gerados utilizando o mecanismo de perturbação apresentado na Figura 23, tido como o primeiro estágio da propagação. O segundo estágio da propagação consiste em aplicar a busca local RIS (Figura 21.a) ao melhor indivíduo gerado no primeiro estágio. A melhor solução encontrada no segundo estágio é adotada para as próximas fases. Pelo que já foi descrito, a fase de propagação considera três importantes parâmetros: λ_{min} , λ_{max} e ω , cujos valores são calibrados na Seção 4.3.

```
//Propagação
Inicialize os parâmetros  $\lambda_{min}, \lambda_{max}, \omega, PN$  e a população  $\Gamma = \emptyset$ 
 $\pi \leftarrow \pi_i, \pi^R \leftarrow \pi_{best}$ 
Calcule o valor de  $d_i$  para  $\pi$  de acordo com a Eq. (67)
Repita//primeiro estágio
     $\pi' \leftarrow$  Faça  $d_i$  inserções aleatórias em  $\pi$ 
     $\Gamma \leftarrow \Gamma \cup \{\pi'\}$ 
Até  $|\Gamma| = \omega$ 
 $\pi \leftarrow$  Selecione a melhor sequencia do conjunto  $\Gamma$ 
Aplique RIS de acordo com a Figura 21.a) considerando  $\pi' \leftarrow \pi'' \leftarrow \pi$ . //Segundo estágio
Retorne  $\pi''$ 
Fim //Propagação
```

Figura 23 - Pseudocódigo da fase de propagação

Fonte: Adaptado de Shao *et al.* (2018a).

4.2.2.2 Refração

Nesta fase, o operador de refração é usado para auxiliar as ondas a escaparem da estagnação e melhorar a diversidade da população. No WWO básico, a área intermediária entre a onda cujo comprimento é zero e a melhor onda da população é explorada nessa fase. Seguindo esta idéia, Shao *et al.* (2018a) propõem a técnica de *Path Relinking* na versão de Vallada e Ruiz (2010), descrita na Figura 24. Para escapar da estagnação, os autores estabelecem o valor de h_{max} , que vai sendo decrescido caso o operador de refração seja ativado, até atingir o valor $h = 0$. Neste caso, a técnica de *Path Relinking* é aplicada à onda.

```

//Refração
Inicialize  $\Gamma = \emptyset$ ,  $\pi \leftarrow \pi_i$  e  $k \leftarrow 0$ 
Repita //primeiro estágio
     $k \leftarrow k + 1$ 
    Se  $\pi(k) \neq \pi_{best}(k)$  Então
        Encontre a posição  $p$  da tarefa  $\pi_{best}(k)$  em  $\pi$ 
        Troque as tarefas nas posições  $p$  e  $k$  em  $\pi$ 
    Se  $\pi \neq \pi_{best}$  Então
         $\Gamma \leftarrow \Gamma \cup \{\pi\}$ 
    FimSe
FimSe
Até  $k = n$ 
Se  $\Gamma \neq \emptyset$  Então
     $\pi \leftarrow$  Melhor sequência de  $\Gamma$ 
Senão
    Selecione aleatoriamente duas posições  $p_1$  e  $p_2$  em  $\pi$ , onde  $p_1 \neq p_2$ . Insira a tarefa da posição  $p_1$  antes da tarefa que está na posição  $p_2$  em  $\pi$ .
FimSe
Retorne  $\pi$ 
Fim //Refração

```

Figura 24 - Pseudocódigo da técnica Path Relinkg usada na fase de refração

Fonte: Autor, 2019.

No procedimento mostrado na Figura 24, a solução inicial, dada pela sequência π e a melhor sequência π_{best} são selecionadas. Então, uma série de movimentos é realizado no intuito de transformar π em π_{best} . A cada movimento, uma nova solução intermediária é obtida e ao final do procedimento, o indivíduo se parece mais com π_{best} do que com π . A solução intermediária que apresenta o mínimo C_{max} dado a restrição do *total_cost* é tida como a melhor sequência. Em casos em que π e π_{best} são muito similares e um único movimento os deixa idênticos, nenhum indivíduo é gerado. Os autores propõem um mecanismo de perturbação, em

que duas posições p_1 e p_2 são aleatoriamente escolhidas e a tarefa que está na posição p_1 é inserida antes da tarefa que ocupa a posição p_2 .

4.2.2.3 Quebra

O operador de quebra tem o papel de intensificar a busca no WWO. Se uma nova solução gerada pelo operador de propagação superar a melhor solução encontrada π_{best} , então, o operador de quebra é ativado. Shao *et al.* (2018a) propõem uma versão do algoritmo *Variable Neighborhood Search* (VNS) para intensificação da busca. Em geral, o método VNS é desenvolvido considerando três decisões: seleção das estruturas de vizinhança, a ordem das estruturas de vizinhança e o operador de troca da aplicação das estruturas. Duas buscas locais, denominadas LS1 e LS2 são adotadas pelos autores. Em LS1, RIS (Figura 21) é adotada como estrutura de vizinhança, com a exceção de que a sequência de referência π^R é diferente. Já LS2, cujo pseudocódigo é apresentado na Figura 25, é similar a estrutura de RIS, porém, utiliza a estrutura de vizinhança de troca entre pares de tarefas.

```
//Procedimento Referenced Permutation Search
Inicialize  $\pi^R \leftarrow$  Sequência aleatoriamente gerada,  $k \leftarrow 1$ ,  $cont \leftarrow 1$ 
 $\pi \leftarrow$  Sequência gerada pelo operador de propagação
Repita
  Troque a tarefa  $\pi^R(k)$  com todas as tarefas após ela em  $\pi$ 
  Se  $\pi(k) \neq \pi_{best}(k)$  Então
    Encontre a posição  $p$  da tarefa  $\pi_{best}(k)$  em  $\pi$ 
     $\pi' \leftarrow$  troque a tarefa  $\pi_{best}(k)$  com a tarefa que resultou no menor  $C_{max}$  dado a restrição de custo total de MP
    Se  $MP_{cost} \geq total\_cost(\pi')$  and  $(C_{max}(\pi') < C_{max}(\pi))$  Então
       $\pi \leftarrow \pi'$  e  $cont \leftarrow 1$ 
    Senão
       $cont \leftarrow cont + 1$ 
    FimSe
     $k \leftarrow (k + 1) \bmod(n)$ 
  FimSe
Até  $cont = n$ 
Retorne  $\pi'$ 
Fim // Procedimento Referenced Permutation Search
```

Figura 25 - Pseudocódigo da Referenced Permutation Search

Fonte: Adaptado de Shao *et al.* (2018a).

As duas buscas locais propostas adotam uma trajetória de busca dinâmica. Para evitar que as estruturas de vizinhança explorem a sequência sempre na mesma ordem, a sequência de referência π^R é aleatoriamente gerada e ambas as estruturas devem mover as tarefas de acordo com esta sequência. Em relação a ordem, LS2 é aplicada primeiro, de modo que forneça área promissoras para LS1. Já em relação a estratégia de troca das estruturas, é adotada a forma genérica. Nesta estratégia, a nova solução obtida na respectiva vizinhança é comparada com a solução incumbente. Se uma melhoria é obtida, então, o processo de busca retorna ao estado inicial e a solução incumbente é substituída pela nova solução. Caso contrário, outra estrutura de vizinhança é considerada. O pseudocódigo do VNS proposto pode ser visto na Figura 26.

```
//VNS
 $\pi \leftarrow$  Sequência gerada pelo operador de propagação e  $l \leftarrow 1$ 
Repita
  Se  $l = 1$  Então
     $\pi' \leftarrow$  LS2( $\pi$ )
  Senão
    Se  $l = 1$  Então
       $\pi' \leftarrow$  LS1( $\pi$ )
  FimSe
  Se  $MP_{cost} \geq total\_cost(\pi')$  and  $(C_{max}(\pi') < C_{max}(\pi))$  Então
     $\pi \leftarrow \pi'$  e  $l \leftarrow 1$ 
  Senão
     $l \leftarrow l + 1$ 
  FimSe
Até  $(l > 2)$ 
Retorne  $\pi$ 
Fim //VNS
```

Figura 26 - Pseudocódigo geral do VNS proposto

Fonte: Adaptado de Shao *et al.* (2018a).

4.2.2.4 Atualização da população

A adaptação da população tem como objetivo manter a qualidade e diversidade das soluções. Neste sentido, se a nova onda criada na fase de propagação apresenta qualidade de solução pior que seu ancestral, porém, é melhor que o pior indivíduo e não é idêntico a nenhum outro indivíduo da população, o pior indivíduo será substituído por esta nova onda. Adicionalmente, para manter a diversidade da população e evitar buscas cíclicas, as novas ondas adicionadas a população são checadas se são idênticas a algum indivíduo da população. Se sim, a nova onda é abandonada.

4.3 EXPERIMENTAÇÕES COMPUTACIONAIS

Esta seção está estruturada da seguinte forma: na Seção 4.3.1 são apresentados os resultados de desempenho das heurísticas construtivas. As próximas seções tratam da calibração parâmetros das metaheurísticas propostas (Seção 4.3.2), análise de desempenho das metaheurísticas em relação aos problemas de pequeno (Seção 4.3.3) e de médio e grande porte (Seção 4.3.4).

4.3.1 Análise de desempenho das heurísticas construtivas

Todas as heurísticas construtivas foram testadas nos dois bancos de problemas de médio e grande porte (20 a 500 tarefas). Como já mencionado, todos os métodos testados nesta seção, tiveram como prioridade a minimização do custo total de manutenção preventiva (MP) (*total_cost*), seguido pela minimização da duração total da programação (C_{max}). Os valores de λ para PF-NEH(x), wPF-NEH(x), PW-NEH(x), PFT-NEH(x), mNEH e suas versões alternativas seguiram as configurações estabelecidas pela literatura. Inicialmente, experimentos computacionais foram conduzidos com as bases referidas, no intuito de calibrar os parâmetros exigidos pelas heurísticas MM, MM-B, mNEH e mNEH_{st}. Para MM e MM-B, o valor de α seguiram as mesmas configurações propostas por Ronconi (2004). Os resultados obtidos pela experimentação computacional de MM podem ser visualizados na Figura 27, que mostra o desempenho do método diante dos cenários YS (a) e RGM (b).

Figura 27 - Calibração do parâmetro α da heurística MM nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Os parâmetros que apresentaram o melhor desempenho, ou seja, $\alpha = 0,25$ para o cenário YS e $\alpha = 0,05$ para o cenário RGM foram adotados como solução da heurística MM. Na Figura 9, apresenta-se a calibração da heurística MM-B para os dois cenários YS (a) e RGM (b).

Figura 28 - Calibração do parâmetro α da heurística MM-B nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Conforme os dados apresentados pela Figura 28, MM-B obteve o melhor desempenho quando α assume o valor 0,15 para o cenário YS e 0 para o cenário RGM. Nas Figura 29 e Figura 30 apresenta-se os desempenhos de mNEH e mNEHst em relação a diferentes valores de λ .

Figura 29 - Calibração do parâmetro λ da heurística mNEH nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Como constatado, mNEH atinge os melhores desempenhos quando $\lambda = 0$ e $\lambda = 0$ e 0,1 para os cenários YS e RGM, respectivamente. Em contrapartida, mNEHst apresenta os desvios relativos médios mais baixos quando λ assume valor 0,4 no cenário YS e 0,6 no cenário RGM.

Figura 30 - Calibração do parâmetro λ da heurística mNEHst nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Os resultados do desempenho das heurísticas em relação ao número de tarefas para os cenários YS e RGM podem ser vistos nas Tabelas 14 e 15, respectivamente.

Tabela 14 - Desempenho das heurísticas construtivas em relação ao custo total de MP para minimização de C_{max} no cenário YS (melhores resultados destacados em negrito) (Continua...)

n	20		50		100		200		500		Média Geral	
Heurística	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo
PF	9,278	0,001	5,220	0,011	3,259	0,118	2,112	0,566	1,503	8,434	4,274	1,826
MM	9,256	0,001	4,899	0,008	2,681	0,052	1,570	0,282	0,736	3,472	3,829	0,763
MM-B	9,540	0,001	4,851	0,009	2,738	0,054	1,574	0,285	0,763	3,516	3,893	0,773
PFE	4,723	0,007	3,108	0,122	2,500	0,698	2,452	5,260	2,789	90,698	3,115	19,357
MME	4,436	0,007	2,876	0,119	2,343	0,633	2,351	4,979	2,748	85,777	2,951	18,303
MME-B	4,420	0,007	2,950	0,119	2,407	0,634	2,369	4,981	2,740	85,813	2,977	18,311
PF-NEH(1)	4,856	0,003	3,414	0,040	2,544	0,282	1,863	1,613	1,421	14,833	2,820	3,354
PF-NEH(1)st	4,540	0,003	3,062	0,040	2,364	0,279	1,807	1,605	1,371	14,740	2,629	3,334
wPF-NEH(1)	4,733	0,007	3,618	0,073	2,530	0,402	1,885	1,697	1,405	13,423	2,834	3,121
wPF-NEH(1)st	4,545	0,007	3,115	0,074	2,392	0,404	1,747	1,665	1,338	13,447	2,627	3,120
PW-NEH(1)	4,701	0,006	3,614	0,072	2,543	0,396	1,863	1,935	1,401	19,396	2,824	4,361
PW-NEH(1)st	4,524	0,006	3,138	0,073	2,370	0,405	1,753	1,956	1,350	19,078	2,627	4,304
PFT-NEH(1)	4,815	0,006	3,433	0,052	2,485	0,302	1,821	1,484	1,412	12,521	2,793	2,873
PFT-NEH(1)st	4,490	0,006	3,056	0,052	2,317	0,307	1,737	1,469	1,352	12,488	2,591	2,864
PFT-RZ(1)	4,590	0,011	3,587	0,122	2,528	1,295	1,835	13,045	1,383	233,968	2,784	49,688
PFT-RZ(1)st	3,775	0,011	3,305	0,131	2,287	1,319	1,727	13,260	1,327	233,572	2,484	49,659
PF-NEH(λ)	4,717	0,044	3,559	0,351	2,642	1,687	1,873	7,959	1,419	66,375	2,842	15,283
PF-NEH(λ)st	3,654	0,044	2,930	0,351	2,384	1,687	1,721	7,959	1,322	66,375	2,402	15,283
WPT-NEH	4,717	0,004	3,168	0,057	2,586	0,502	2,495	4,062	2,766	72,277	3,146	15,380
WPT-NEHst	3,654	0,004	2,167	0,054	1,490	0,499	1,315	4,705	1,576	76,062	2,041	16,265
PF-NEH(2)	3,381	0,006	2,628	0,080	2,021	0,564	1,480	3,225	1,174	29,666	2,137	6,708
PF-NEH(2)st	3,275	0,007	2,427	0,080	1,837	0,558	1,458	3,211	1,162	29,481	2,032	6,667
wPF-NEH(2)	3,240	0,015	2,804	0,147	1,962	0,805	1,499	3,394	1,149	26,845	2,131	6,241
wPF-NEH(2)st	3,252	0,015	2,531	0,149	1,888	0,808	1,411	3,331	1,137	26,894	2,044	6,239

Tabela 14 - Desempenho das heurísticas construtivas em relação ao custo total de MP para minimização de C_{max} no cenário YS (melhores resultados destacados em negrito) (Conclusão).

n	20		50		100		200		500		Média Geral	
Heurística	DR	Tempo	DR	Tempo	DR	Tempo	Heurística	DR	Tempo	DR	Tempo	DR
PW-NEH(2)	3,218	0,012	2,834	0,144	1,977	0,792	1,477	3,871	1,146	38,793	2,131	8,722
PW-NEH(2)st	3,256	0,012	2,529	0,145	1,902	0,810	1,417	3,912	1,144	38,156	2,049	8,607
PFT-NEH(2)	3,383	0,011	2,624	0,104	1,995	0,605	1,462	2,969	1,160	25,043	2,125	5,746
PFT-NEH(2)st	3,229	0,012	2,443	0,104	1,848	0,614	1,416	2,939	1,124	24,977	2,012	5,729
PFT-RZ(2)	3,268	0,021	2,703	0,243	1,941	2,589	1,452	26,091	1,126	467,937	2,098	99,376
PFT-RZ(2)st	2,950	0,022	2,587	0,263	1,803	2,637	1,395	26,520	1,101	467,145	1,967	99,317
PF-NEH(5)	2,029	0,016	1,821	0,199	1,439	1,411	1,111	8,063	0,918	74,165	1,464	16,771
PF-NEH(5)st	1,985	0,017	1,673	0,200	1,328	1,394	1,051	8,027	0,894	73,701	1,386	16,668
wPF-NEH(5)	1,953	0,037	1,876	0,366	1,418	2,012	1,073	8,485	0,908	67,113	1,446	15,603
wPF-NEH(5)st	1,964	0,037	1,837	0,372	1,381	2,020	1,041	8,327	0,873	67,236	1,419	15,598
PW-NEH(5)	1,957	0,031	1,866	0,360	1,402	1,981	1,063	9,677	0,899	96,982	1,437	21,806
PW-NEH(5)st	1,932	0,031	1,829	0,363	1,353	2,026	1,047	9,780	0,889	95,389	1,410	21,518
PFT-NEH(5)	2,008	0,034	1,784	0,259	1,411	1,353	1,089	6,897	0,897	61,874	1,438	14,083
PFT-NEH(5)st	1,912	0,030	1,668	0,259	1,334	1,535	1,052	7,346	0,877	62,442	1,369	14,322
PFT-RZ(5)	2,087	0,054	1,821	0,609	1,302	6,473	1,048	65,227	0,871	1169,842	1,426	248,441
PFT-RZ(5)st	1,812	0,055	1,794	0,657	1,258	6,593	1,010	66,299	0,842	1167,861	1,343	248,293
mNEH	4,328	0,004	2,846	0,055	2,247	0,470	2,122	4,192	2,354	75,923	2,779	16,129
mNEHst	3,935	0,004	2,284	0,065	1,530	0,647	1,439	4,421	1,688	78,157	2,175	16,659

Fonte: Autor 2019.

Como pode ser observado, as versões de heurísticas que admitem até 5 iterações apresentaram os melhores desempenhos nas duas políticas avaliadas, YS e RGM. Pode-se notar que PFT-RZ(5)st e PFT-NEH(5)st obtiveram os melhores desempenhos em relação a qualidade de solução.

Tabela 15 - Desempenho das heurísticas construtivas em relação ao custo total de MP para minimização de C_{max} no cenário RGM (melhores resultados destacados em negrito) (Continua...)

n	20		50		100		200		500		Média Geral	
Heurística	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo
PF	9,278	0,001	5,220	0,006	3,259	0,042	2,112	0,300	1,503	4,520	4,274	0,974
MM	9,256	0,001	4,899	0,008	2,681	0,044	1,570	0,263	0,736	2,707	3,829	0,605
MM-B	9,540	0,001	4,851	0,008	2,738	0,043	1,574	0,251	0,763	2,636	3,893	0,588
PFE	4,723	0,007	3,108	0,122	2,500	0,698	2,452	5,260	2,789	85,071	3,115	18,232
MME	4,436	0,007	2,876	0,119	2,343	0,633	2,351	4,979	2,748	80,149	2,951	17,177
MME-B	4,420	0,007	2,950	0,119	2,407	0,634	2,369	4,981	2,740	80,183	2,977	17,185
PF-NEH(1)	4,856	0,003	3,414	0,040	2,544	0,282	1,863	1,613	1,421	14,833	2,820	3,354
PF-NEH(1)st	4,540	0,003	3,062	0,038	2,364	0,265	1,807	1,525	1,371	14,003	2,629	3,167

Tabela 15 - Desempenho das heurísticas construtivas em relação ao custo total de MP para minimização de C_{max} no cenário RGM (melhores resultados destacados em negrito) (Conclusão.)

n	20		50		100		200		500		Média Geral	
Heurística	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo	DR	Tempo
wPF-NEH(1)	4,733	0,007	3,618	0,070	2,530	0,382	1,885	1,612	1,405	12,751	2,834	2,965
wPF-NEH(1)st	4,545	0,007	3,115	0,071	2,392	0,384	1,747	1,582	1,338	12,775	2,627	2,964
PW-NEH(1)	4,701	0,006	3,614	0,068	2,543	0,376	1,863	1,839	1,401	18,427	2,824	4,143
PW-NEH(1)st	4,524	0,006	3,138	0,069	2,370	0,385	1,753	1,858	1,350	18,124	2,627	4,088
PFT-NEH(1)	4,815	0,005	3,433	0,049	2,485	0,287	1,821	1,410	1,412	11,895	2,793	2,730
PFT-NEH(1)st	4,490	0,006	3,056	0,049	2,317	0,292	1,737	1,396	1,352	11,864	2,591	2,721
PFT-RZ(1)	4,590	0,009	3,587	0,106	2,528	0,964	1,835	8,711	1,383	169,006	2,784	35,759
PFT-RZ(1)st	3,775	0,009	3,305	0,106	2,287	0,978	1,727	8,826	1,327	172,646	2,484	36,513
PF-NEH(λ)	4,717	0,002	3,559	0,034	2,642	0,190	1,873	1,048	1,419	10,690	2,842	2,393
PF-NEH(λ)st	3,654	0,002	2,930	0,034	2,384	0,190	1,721	1,047	1,322	10,602	2,402	2,375
WPT-NEH	4,717	0,003	3,168	0,037	2,586	0,331	2,495	2,828	2,766	47,901	3,146	10,220
WPT-NEH(st)	3,654	0,003	2,167	0,037	1,490	0,332	1,315	2,845	1,576	48,203	2,041	10,284
PF-NEH(2)	3,381	0,006	2,628	0,080	2,021	0,564	1,480	3,225	1,174	29,666	2,137	6,708
PF-NEH(2)st	3,275	0,006	2,427	0,076	1,837	0,530	1,458	3,050	1,162	28,007	2,032	6,334
wPF-NEH(2)	3,240	0,014	2,804	0,139	1,962	0,765	1,499	3,224	1,149	25,503	2,131	5,929
wPF-NEH(2)st	3,252	0,014	2,531	0,141	1,888	0,768	1,411	3,164	1,137	25,550	2,044	5,927
PW-NEH(2)	3,218	0,012	2,834	0,137	1,977	0,753	1,477	3,677	1,146	36,853	2,131	8,286
PW-NEH(2)st	3,256	0,012	2,529	0,138	1,902	0,770	1,417	3,716	1,144	36,248	2,049	8,177
PFT-NEH(2)	3,383	0,011	2,624	0,099	1,995	0,575	1,462	2,820	1,160	23,791	2,125	5,459
PFT-NEH(2)st	3,229	0,012	2,443	0,099	1,848	0,583	1,416	2,792	1,124	23,728	2,012	5,443
PFT-RZ(2)	3,268	0,018	2,703	0,211	1,941	1,928	1,452	17,421	1,126	338,012	2,098	71,518
PFT-RZ(2)st	2,950	0,017	2,587	0,213	1,803	1,956	1,395	17,652	1,101	345,292	1,967	73,026
PF-NEH(5)	2,029	0,016	1,821	0,199	1,439	1,411	1,111	8,063	0,918	74,165	1,464	16,771
PF-NEH(5)st	1,985	0,016	1,673	0,190	1,328	1,325	1,051	7,626	0,894	70,016	1,386	15,835
wPF-NEH(5)	1,953	0,035	1,876	0,348	1,418	1,911	1,073	8,061	0,908	63,757	1,446	14,823
wPF-NEH(5)st	1,964	0,035	1,837	0,354	1,381	1,919	1,041	7,911	0,873	63,874	1,419	14,819
PW-NEH(5)	1,957	0,030	1,866	0,342	1,402	1,882	1,063	9,193	0,899	92,133	1,437	20,716
PW-NEH(5)st	1,932	0,029	1,829	0,345	1,353	1,925	1,047	9,291	0,889	90,620	1,410	20,442
PFT-NEH(5)	2,008	0,027	1,784	0,247	1,411	1,437	1,089	7,051	0,897	59,476	1,438	13,648
PFT-NEH(5)st	1,912	0,029	1,668	0,246	1,334	1,458	1,052	6,979	0,877	59,319	1,369	13,606
PFT-RZ(5)	2,087	0,045	1,821	0,529	1,302	4,820	1,048	43,553	0,871	845,029	1,426	178,795
PFT-RZ(5)st	1,812	0,044	1,794	0,531	1,258	4,890	1,010	44,130	0,842	863,230	1,343	182,565
mNEH	4,328	0,003	2,846	0,055	2,247	0,504	2,122	4,845	2,354	113,178	2,779	23,717
mNEHst	3,935	0,003	2,284	0,057	1,530	0,495	1,439	4,799	1,688	113,138	2,175	23,698

Fonte: Autor 2019.

Nas Figuras 31 e 32, são evidenciadas as diferenças de desempenho das heurísticas em relação aos cenários YS e RGM, respectivamente.

Figura 31 – Gráfico de médias do desvio relativo ao nível de 5% de significância de heurísticas construtivas para minimização do custo e C_{max} no cenário YS

Fonte: Autor, 2019.

Figura 32 – Gráfico de médias do desvio relativo ao nível de 5% de significância de heurísticas construtivas para minimização do custo e C_{max} no cenário RGM

Fonte: Autor, 2019.

Pela análise dos gráficos apresentados, constata-se no cenário YS, onde se admite intervalos entre MP's menos espaçados, que as heurísticas que utilizam até 5 iterações apresentam diferença estatística significativa em relação ao restante dos métodos avaliados. De modo geral, as versões alternativas propostas obtiveram resultados superiores do que as versões originais. Já para o cenário RGM, observa-se uma diferença maior nos desempenhos de PFT-RZ(5) e PFT-RZ(5)st em comparação com as outras versões que admitem 5 iterações, apesar de não apresentarem diferença estatística significativa. É mais evidente também que as heurísticas, de um modo geral, tendem a obter desempenhos mais próximos, uma vez que,

quanto mais espaçadas os intervalos entre MP's, um número menor de atividades de MP's são realizados. Por consequência, a função de custo total de MP sofre uma variação menor que em relação ao cenário YS. Nas Tabelas 16 e 17 são mostrados os resultados do teste de honestidade de Tukey ao nível de 5% de significância para identificação de subconjuntos homogêneos dos métodos avaliados.

Tabela 16 - Teste de honestidade de Tukey ao nível de 5% de significância para cenário YS (Continua...)

Heurística	1	2	3	4	5	6	7	8	9	10
PFT-RZ(5)st	1,343									
PFT-NEH(5)st	1,369									
PF-NEH(5)st	1,386									
PW-NEH(5)st	1,410									
wPF-NEH(5)st	1,419									
PFT-RZ(5)	1,426									
PW-NEH(5)	1,437									
PFT-NEH(5)	1,438									
wPF-NEH(5)	1,446									
PF-NEH(5)	1,464									
PFT-RZ(2)st		1,967								
PFT-NEH(2)st		2,012								
PF-NEH(2)st		2,032								
WPT-NEHst		2,041								
wPF-NEH(2)st		2,044								
PW-NEH(2)st		2,049								
PFT-RZ(2)		2,098								
PFT-NEH(2)		2,125								
PW-NEH(2)		2,131								
wPF-NEH(2)		2,131								
PF-NEH(2)		2,137								
mNEHst		2,175	2,175							
PF-NEH(λ)st			2,402	2,402						
PFT-RZ(1)st				2,484						
PFT-NEH(1)st				2,591	2,591					
PW-NEH(1)st				2,627	2,627	2,627				
wPF-NEH(1)st				2,627	2,627	2,627				
PF-NEH(1)st				2,629	2,629	2,629				
mNEH					2,779	2,779	2,779			
PFT-RZ(1)					2,784	2,784	2,784			
PFT-NEH(1)					2,793	2,793	2,793			
PF-NEH(1)					2,820	2,820	2,820			
PW-NEH(1)					2,824	2,824	2,824			
wPF-NEH(1)						2,834	2,834			

Tabela 16 - Teste de honestidade de Tukey ao nível de 5% de significância para cenário YS (Conclusão.)

Heurística	1	2	3	4	5	6	7	8	9	10
PF-NEH(λ)						2,842	2,842			
MME							2,951	2,951		
MME-B							2,977	2,977		
PFE								3,115		
WPT-NEH								3,146		
MM									3,829	
MM-B									3,893	
PF										4,274
Significância	0,996	0,193	0,074	0,076	0,050	0,141	0,296	0,324	1	1

Fonte: Autor, 2019.

Para o cenário YS, o teste de honestidade de Tukey identificou 10 subgrupos, nas quais são agrupadas por médias do desvio relativo que não apresentam diferenças estatísticas significativas. Cada subconjunto é organizado em ordem crescente do desvio relativo médio.

Tabela 17 - Teste de honestidade de Tukey ao nível de 5% de significância para cenário RGM (Continua...)

Heurística	1	2	3	4	5	6	7	8	9	10	11	12	13	14
PFT-RZ(5)st	1,690													
PFT-RZ(5)	1,726													
PFT-NEH(5)st	2,029													
PW-NEH(5)st	2,046													
wPF-NEH(5)st	2,050													
PF-NEH(5)st	2,053													
PFT-NEH(5)	2,090													
PW-NEH(5)	2,100	2,100												
PF-NEH(5)	2,107	2,107												
wPF-NEH(5)	2,112	2,112												
PFT-RZ(2)st		2,375	2,375											
PFT-RZ(2)			2,490	2,490										
WPT-NEHst				2,694	2,694									
PFT-NEH(2)st				2,751	2,751	2,751								
PF-NEH(2)st				2,763	2,763	2,763								
PW-NEH(2)st					2,774	2,774								
mNEHst					2,782	2,782								
wPF-NEH(2)st					2,786	2,786								

Tabela 17 - Teste de honestidade de Tukey ao nível de 5% de significância para cenário RGM (Conclusão.)

Heurística	1	2	3	4	5	6	7	8	9	10	11	12	13	14
PFT-NEH(2)						2,869	2,869							
PF-NEH(2)						2,877	2,877							
PW-NEH(2)						2,886	2,886							
wPF-NEH(2)						2,894	2,894							
PFT-RZ(1)st							2,993	2,993						
PFT-RZ(1)								3,197	3,197					
MME-B								3,231	3,231					
MME								3,234	3,234					
PF-NEH(λ)st									3,302	3,302				
mNEH									3,378	3,378	3,378			
PF-NEH(1)st									3,386	3,386	3,386			
WPT-NEH									3,404	3,404	3,404			
PFT-NEH(1)st									3,408	3,408	3,408			
wPF-NEH(1)st									3,426	3,426	3,426			
PW-NEH(1)st									3,429	3,429	3,429			
PFE									3,442	3,442	3,442			
PFT-NEH(1)										3,564	3,564			
PW-NEH(1)											3,588			
wPF-NEH(1)											3,604			
PF-NEH(1)											3,619	3,619		
PF-NEH(λ)											3,637	3,637	3,637	
MM												3,891	3,891	
MM-B													3,906	
PF														5,193
Significância	1	1	0,071	1	0,076	0,730	0,259	0,276	0,242	0,123	0,141	0,08	0,093	1

Fonte: Autor, 2019.

Já para o cenário RGM, conforme pode ser visto na Tabela 17, constata-se que o teste de Tukey identificou 14 subconjuntos diferentes. Cabe destacar que uma mesma heurística pode pertencer a mais de um subconjunto. Diferente do que ocorre em YS, PFT-RZ(5) e PFT-RZ(5)st se diferenciam estatisticamente do restante das heurísticas que admitem cinco iterações. Para heurísticas que adotam o mecanismo de inserção NEH, WPT-NEH_{st} obteve o melhor desempenho. Entretanto, nota-se pela Tabela 17 que versões das heurísticas que admitem cinco

iterações, em especial, PFT-NEH_{st}(2) PFT-NEH(2) e PW-NEH(2), que pertencem ao mesmo subgrupo de WPT-NEH_{st} em ambos os cenários, apresentam desempenhos similares sem diferença estatística significativa, em tempo computacional menor.

Em relação ao número de tarefas (n), de modo geral, percebe-se que heurísticas com mecanismos similares, como é o caso das heurísticas PFT-NEH(x), PF-NEH(x), wPF-NEH(x) e PW-NEH(x) tendem a obter desempenhos similares em todos os grupos de tarefas. É visível que o desvio médio relativo das heurísticas tende a diminuir a medida que o número de tarefas aumenta para a maioria das heurísticas, com exceção daquelas que adotaram o mecanismo de inserção de NEH (Figura 9).

Figura 33 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação ao número de tarefas no cenário YS

Fonte: Autor, 2019.

Como constatado nas Figuras 31 e 32, pode-se constatar que as heurísticas tendem a obter desempenhos mais próximos no cenário RGM. Em ambos os cenários, pode-se notar que as heurísticas MM, MM-B e PF apresentam melhoria significativa a medida que n aumenta.

Figura 36 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação ao número de máquinas no cenário RGM

Fonte: Autor, 2019.

Já em relação aos tempos de *setup*, pode-se constatar pelas Figuras 37 e 38 que as heurísticas que adotam o mecanismo de inserção NEH apresentam desempenhos médios muito similares em todas as faixas de tempos de *setup* consideradas. Em ambos os cenários, pode-se perceber que os pontos médios de desempenho para cada variação do *setup* em WPT-NEH_{st}, mNEH_{st}, mNEH, MME, MME-B, PFE WPT-NEH tendem a se sobrepor ou estarem muito próximos uns dos outros.

Figura 37 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos tempos de *setup* no cenário YS

Fonte: Autor, 2019.

O restante das heurísticas tende a gerar resultados médios mais distantes da melhor solução encontrada com o aumento do tempo de *setup*. Em ambos os cenários, as versões alternativas propostas, apresentaram resultados superiores em comparação com as heurísticas adaptadas ao problema. Adicionalmente, pode-se constatar que na média geral, WPT-NEH_{st} e mNEH_{st} obtêm desvios relativos médios menores e equiparáveis as heurísticas que admitem até cinco iterações nos cenários de *setup* de 100% e 125% dos tempos de processamento, respectivamente.

Figura 38 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos tempos de *setup* no cenário RGM

Fonte: Autor, 2019.

Figura 39 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos níveis de manutenção preventiva no cenário YS

Fonte: Autor, 2019.

Pelos níveis de manutenção (l), nota-se que todas as heurísticas tendem a apresentar desvios relativos menores a medida que l aumenta. Conforme pode ser visualizado pelas Figuras 39 e 40, o desempenho de cada heurística que admite até cinco iterações tende a estar mais próximo com o aumento de l . Já para heurísticas que adotam o mecanismo de inserção NEH há uma distinção maior no desempenho de cada método entre os níveis de MP.

Figura 40 - Desempenho das heurísticas para minimização do custo total de MP e C_{max} em relação aos níveis de manutenção preventiva no cenário RGM

Fonte: Autor, 2019.

Pelos resultados apresentados anteriormente, PFT-RZ(5)st, PFT-RZ(5) e PFT-NEH(5)st apresentam os melhores resultados em qualidade de solução. Entretanto, convém também ponderar o custo para se encontrar esta solução, ou seja, o tempo computacional médio dispendido. Para avaliar o *trade-off* entre eficácia e eficiência, foram elaborados os gráficos apresentados nas Figuras 41 e 42.

Figura 42 - Desempenho das heurísticas do *trade-off* entre desvio relativo e tempo computacional no cenário RGM
Fonte: Autor, 2019.

Neste estudo, considerou-se o uso de PFT-NEH(5)st para geração da solução inicial das metaheurísticas adaptadas para os dois cenários estudados. Apesar de obter desempenho inferior em qualidade de solução em comparação à PFT-RZ(5)st, a heurística escolhida consegue gerar soluções próximas a do melhor método em tempo significativamente menor, em especial para instâncias de grande porte.

4.3.2 Calibração de parâmetros das metaheurísticas IG E DWWO

4.3.2.1 Parametrização de *Iterated Greedy* (IG)

Para a análise de parâmetros das duas versões de *Iterated Greedy* (IG) foram utilizadas as instâncias geradas para ajustamento dos parâmetros, conforme descrito no Capítulo 3. Os valores de d , jP e τP seguiram o proposto por Tasgetiren *et al.* (2017), ou seja, $d = \{4, 5, 6, 7, 8\}$, $jP = \{0, 1, 0,1, 0,01, 0,001\}$ e $\tau P = \{0,1, 0,2, 0,3, 0,4\}$. O *design* de experimentos resultou em um total de $jP \times d \times \tau P = 100$ versões de IG com diferentes parâmetros. Para cada combinação de jP , d e τP , cada versão de IG foi rodada três vezes de forma independente e o melhor valor de C_{max} encontrado entre os três testes foi adotado como solução final da dada combinação.

Utilizou-se como critério de parada um tempo computacional de $n \times m \times l \times 30$ milisegundos. Uma análise de variância (ANOVA) foi aplicada aos dados obtidos, tendo como variável dependente o desvio relativo das soluções encontradas e como variáveis independentes d , tP , jP e as interações entre elas, ou seja, $d*\tau P$, $d*jP$ e $tP*jP$. Considerou-se um nível de significância de 5% para a hipótese nula de que existe diferença significativa dentro de cada subconjunto das variáveis independentes. Se o valor de p for menor que 0,05, pode-se considerar que a diferença estatística das médias dos desvios relativos entre as variáveis independentes é significativa, caso contrário, as médias não se diferem estatisticamente. Os resultados da análise da ANOVA para o cenário YS podem ser vistos na Tabela 18.

Tabela 18 - ANOVA de duas vias para parâmetros do IG no cenário YS

Fatores	Soma dos quadrados	Graus de liberdade	Quadrado médio	Estatística F	Valor de p
d	18705,273	4	4676,318	336,090	0,000
τP	15,915	3	5,305	0,381	0,767
jP	1658,169	4	414,542	29,793	0,000
$d*\tau P$	12,566	12	1,047	0,075	1,000
$d*jP$	76,401	16	4,775	0,343	0,993
$\tau P*jP$	18,282	12	1,524	0,109	1,000

Fonte: Autor, 2019.

Para o cenário YS, o teste indica que há significância estatística dentro das variáveis independentes d e jP , entretanto, não há diferença estatística para τP e para as interações entre os parâmetros. Para avaliar as diferenças estatísticas dentro de cada grupo, um teste de honestidade de Tukey foi aplicado para os fatores d e jP .

Tabela 19 – Teste de honestidade de Tukey ao nível de 5% de significância para IG no cenário YS

d	N	Subconjuntos				
		1	2	3	4	5
4	32.000	2,375				
5	32.000		2,742			
6	32.000			3,014		
7	32.000				3,212	
8	32.000					3,322
Significância		1,000	1,000	1,000	1,000	1,000

Fonte: Autor, 2019.

Conforme observado na Tabela 19, o teste de honestidade de Tukey identificou cinco subconjuntos dentro do fator d , ou seja, há diferença estatística entre todos os valores de d no nível de 5% de significância. Para o fator jP , dois subgrupos são identificados, ou seja, não foi observado diferença estatística entre os valores de jP adotados, os valores de $jP = \{0; 0,001\}$ obtiveram médias de 2,799% e 2,819% e o subconjunto $jP = \{0,1; 1; 0,1\}$ corresponderam a médias iguais a 3,007%, 3,0157% e 3,026%, respectivamente, ao nível de 5% de significância.

Tabela 20 - ANOVA de duas vias para parâmetros do IG no cenário RGM

Fatores	Soma dos quadrados	Graus de liberdade	Quadrado médio	Estatística F	Valor de p
d	14206,504	4	3551,626	269,390	0,000
jP	491,653	3	163,884	12,431	0,000
τP	3522,679	4	880,670	66,799	0,000
$d * jP$	753,356	12	62,780	4,762	0,000
$d * \tau P$	133,208	16	8,325	0,631	0,861
$jP * \tau P$	28,704	12	2,392	0,181	0,999

Fonte: Autor, 2019.

Os resultados da ANOVA para o IG no cenário RGM apresentou diferenças em relação ao cenário YS. Todos os fatores considerados, d , jP e τP e a interação entre $d * jP$ apresentaram diferença estatística significativa ao nível de 5% de significância. Pela Tabela 21 pode-se observar que, assim como em YS, o teste de Tukey identificou cinco subgrupos para o fator d , sendo o subgrupo de 4 tarefas removidas o que apresentou a menor média.

Tabela 21 – Teste de honestidade de Tukey ao nível de 5% de significância para IG no cenário RGM

d	N	Subconjuntos				
		1	2	3	4	5
4	32.000	2,484				
5	32.000		2,830			
6	32.000			3,052		
7	32.000				3,209	
8	32.000					3,324
Significância		1,000	1,000	1,000	1,000	1,000

Fonte: Autor, 2019.

Em relação ao fator τP , o teste de Tukey identificou dois subconjuntos distintos, sendo $\tau P = \{0,2; 0,3; 0,1\}$ com médias iguais a 2,939%, 2,951% e 2,954%, respectivamente e $\tau P = \{0,4\}$ com média igual a 3,075%. Para o fator jP , foram também identificados dois subgrupos,

sendo $jP = \{0; 0,001\}$ com médias iguais a 2,788% e 2,809%, respectivamente e $jP = \{1; 0,01; 0,1\}$ com médias iguais a 3,089%, 3,105% e 3,108%, respectivamente. Uma ANOVA foi aplicada para a interação entre d e jP . Pode-se constatar que a diferença estatística entre os valores de jP dentro de cada fator d ocorre quando $d = 4$ e 5. Os resultados da interação entre $d = \{4,5\}$ e $jP = \{0,1, 0,2, 0,3, 0,4\}$ podem ser vistos na Tabela 22.

Tabela 22 – Teste de honestidade de Tukey ao nível de 5% de significância para a interação $d * jP$ no cenário RGM

d	(I) jP	(J) jP	Diferença Média (I-J)	Std. Error	Sig. ^b
4	0,1	0,2	-0,039	0,057	0,494
		0,3	-0,038	0,057	0,509
		0,4	-0,402*	0,057	0,000
	0,2	0,1	0,039	0,057	0,494
		0,3	0,001	0,057	0,982
		0,4	-0,362*	0,057	0,000
	0,3	0,1	0,038	0,057	0,509
		0,2	-0,001	0,057	0,982
		0,4	-0,364*	0,057	0,000
	0,4	0,1	0,402*	0,057	0,000
		0,2	0,362*	0,057	0,000
		0,3	0,364*	0,057	0,000
5	0,1	0,2	0,026	0,057	0,645
		0,3	0,002	0,057	0,969
		0,4	-0,237*	0,057	0,000
	0,2	0,1	-0,026	0,057	0,645
		0,3	-0,024	0,057	0,673
		0,4	-0,264*	0,057	0,000
	0,3	0,1	-0,002	0,057	0,969
		0,2	0,024	0,057	0,673
		0,4	-0,239*	0,057	0,000
	0,4	0,1	0,237*	0,057	0,000
		0,2	0,264*	0,057	0,000
		0,3	0,239*	0,057	0,000

* A diferença é significativa ao nível de 5% de significância

Fonte: Autor, 2019.

Pode-se visualizar na Tabela 22 que dentro dos conjuntos $d = 4$, há diferenças estatísticas entre $jP = \{(0,1, 0,4), (0,2, 0,4), (0,3, 0,4)\}$ ao nível de significância de 5%. No geral, o

algoritmo atinge o melhor desempenho quando $d = 4$ e $jP = 0,1$. Os resultados da média do desvio relativo dentro de cada fator podem ser vistos das Figuras 43 à 45.

Figura 43 - Desempenho de IG em relação ao parâmetro d nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Em relação ao número de tarefas removidas (d), constata-se que para ambos os cenários, o IG atinge o melhor desempenho quando $d = 4$. A Figura 43 também sugere que quanto maior o número de tarefas removidas, mais o algoritmo tende a fornecer soluções próximas a solução dada como *input* na etapa de inicialização, devido ao aumento do desvio relativo médio. O comportamento de IG em relação ao valor de τP para os cenários YS e RGM pode ser visto na Figura 44 pelos gráficos a) e b), respectivamente.

Figura 44 - Desempenho de IG em relação ao parâmetro τP nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Pode-se notar que o aumento do valor de τP resulta em temperaturas T com valores maiores e, conseqüentemente, aumenta-se as chances de aceitar uma solução encontrada para a próxima iteração. Neste sentido, os gráficos a) e b) apresentados nas Figura 44, assim como os testes de Tukey apresentados anteriormente, sugerem que para o cenário YS e RGM, τP apresenta a menor média do desvio relativo quando assume os valores 0,3 e 0,2, respectivamente.

Figura 45 - Desempenho de IG em relação ao parâmetro jP nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Como evidenciado na Figura 45 nos gráficos a) e b), o IG atinge as menores médias do desvio relativo quando $jP = 0$ e 0,001. Ressalta-se que com o valor de $jP = 0$, a versão IG_RIS é utilizada, uma vez que a busca local BestSwap não é ativada.

Pelos resultados apresentados anteriormente, sugere-se a utilização de diferentes parâmetros para as instâncias de problemas adotadas. Portanto, para o cenário YS, $d = 4$, $jP = 0$ e 0,001 e $\tau P = 0,3$ e $d = 4$, $jP = 0$ e 0,001 e $\tau P = 0,2$ para o cenário RGM.

4.3.2.2 Parametrização de *Discrete Water Wave Optimization Algorithm* (DWWO)

A análise de parâmetros de DWWO seguiu as configurações sugeridas por Shao *et al.* (2018a) para cada fator. Neste caso, a população $pop \in \{5, 15, 25, 35\}$, $hmax \in \{10, 20, 30\}$, $\omega \in \{20, 30, 40\}$ e $\lambda_{min}; \lambda_{max} \in \{(1 \text{ e } 2), (1 \text{ e } 3), (1 \text{ e } 4), (2 \text{ e } 3), (2 \text{ e } 4)\}$, o que resulta em um total de $4 \times 3 \times 3 \times 5 = 180$ configurações do algoritmo proposto. Cada configuração foi rodada

três vezes de forma independente para o banco de instâncias de problemas de ajuste de parâmetros. O melhor resultado dos três experimentos foi adotado como a solução para a dada configuração. O critério de parada foi estabelecido pelo tempo computacional, sendo calibrado de acordo com $n \times m \times l \times 30$ milissegundos. Como já pontuado, PFT-NEH(5)st foi utilizado para gerar o primeiro indivíduo da população. Os resultados médios do desempenho do algoritmo em relação aos parâmetros adotados no cenário YS podem ser vistos nas Figuras 46 e 47.

Figura 46 - Desempenho do algoritmo no cenário YS em relação aos parâmetros: a) população e b) h_{max}

Fonte: Autor, 2019.

De acordo com a média do desvio relativo, os gráficos apresentados na Figura 47.a) e b) mostram que em relação a população e o parâmetro h_{max} , o algoritmo atinge o melhor desempenho médio para $pop = 25$ e $h_{max} = 20$, respectivamente.

Figura 47 - Desempenho do algoritmo no cenário YS em relação aos parâmetros: a) ω e b) λ

Fonte: Autor, 2019.

Em relação aos parâmetros ω e λ , constata-se pela Figura 47 por meio dos gráficos a) e b), respectivamente, que o menor desvio relativo médio é obtido quando $\omega = 30$ e $\lambda = (1,2)$. Uma ANOVA foi realizada para verificar a diferença estatística dos fatores e das interações entre os fatores. Os resultados da análise para o cenário YS pode ser visto na Tabela 23.

Tabela 23 – ANOVA ao nível de 5% de significância para DWWO no cenário YS

Fatores	Soma dos Quadrados	Graus de liberdade	Quadrado médio	Estatística F	Significância
pop	126,112	3	42,037	4,778	0,002
hmax	5,412	2	2,706	0,308	0,735
ω	39,873	2	19,936	2,266	0,104
λ	596,671	4	149,168	16,956	0,000
pop * hmax	5,585	6	0,931	0,106	0,996
pop * ω	194,360	6	32,393	3,682	0,001
pop * λ	434,537	12	36,211	4,116	0,000
hmax * ω	12,134	4	3,033	0,345	0,848
hmax * λ	46,531	8	5,816	0,661	0,726
ω * λ	576,827	8	72,103	8,196	0,000

Fonte: Autor, 2019.

Como pode ser constatado, o parâmetro pop e λ e as interações pop* ω , pop* λ e ω * λ apresentam diferença estatística significativa, uma vez que o valor de p apresenta valor menor que 0,05. Pelo teste de honestidade de Tukey pode-se identificar dois subconjuntos de médias, $pop = \{25,15,35\}$ com médias iguais a 2,357%, 2,359% e 2,361%, respectivamente, e um subconjunto contendo $pop = 5$ com média igual a 2,407%. Neste sentido, pode-se constatar que $pop = 25$ é o parâmetro que apresenta a menor média do desvio relativo, embora não apresente diferença estatística com $pop = 15$ e 35. Já para o parâmetro λ , foram identificados 3 subconjuntos, $\lambda = \{(1,2)\}$ com média de 2,289%, $\lambda = \{(1,4), (1,3), (2,4)\}$ com respectivas médias correspondentes a 2,368%, 2,372% e 2,403% e $\lambda = \{(2,4) \text{ e } (2,3)\}$ onde o primeiro elemento apresenta média de 2,403% e o segundo elemento, média de 2,423%.

Figura 48 - Interação entre os parâmetros pop * ω no cenário YS

Fonte: Autor, 2019.

O gráfico apresentado na Figura 48 mostra a variação da média do desvio relativo para diferentes valores de ω em cada valor de pop adotado. Conforme pode ser constatado, o ponto que obteve o menor valor ocorreram na interação entre pop = 25 e $\omega = \{20\}$.

Pela Figura 49 pode ser constatado que o parâmetro $\lambda = (1,2)$ obtém os menores médias do desvio relativo na interação com os valores de pop, principalmente, quando pop = 5 e 25. Pela análise gráfica, o ponto $\lambda = (1,2)$ e pop = 25 apresenta o menor desvio relativo.

Figura 49 - Interação entre pop * λ no cenário YS

Fonte: Autor, 2019.

Figura 50 - Interação entre ω * λ no cenário YS

Fonte: Autor, 2019.

Para a interação entre $\omega * \lambda$ nota-se que os parâmetros $\lambda = (1,2)$ e $(1,3)$ obtiveram as menores médias para os valores de ω considerados. Dentre todos os pontos plotados no gráfico apresentado na Figura 50, o ponto $\omega = 40$ e $\lambda = (1,2)$ apresentou a menor média. Pelas análises apresentadas anteriormente, é sugerido como parâmetros para o cenário YS, $\text{pop} = 25$, $\lambda = (1,2)$, $\text{hmax} = 20$ e $\omega = 30$. As análises gráficas do desempenho de DWWO frente aos parâmetros pop , hmax , λ e ω podem ser vistos nas Figuras 51 e 52.

Figura 51 - Desempenho do algoritmo no cenário RGM em relação aos parâmetros: a) população e b) hmax

Fonte: Autor, 2019.

Pelos gráficos a) e b) apresentados na Figura 51 pode-se constatar que o algoritmo atinge os menores desvios relativos médio quando $\text{pop} = 25$ e $\text{hmax} = 20$, similar aos resultados obtidos no cenário YS.

Figura 52 - Desempenho do algoritmo no cenário RGM em relação aos parâmetros: a) ω e b) λ

Fonte: Autor, 2019.

Pelos gráficos a) e b), dispostos na Figura 52, é evidenciado que o algoritmo obteve os menores desvios relativos médios para valores $\omega = 40$ e $\lambda = (1, 2)$. Na Tabela 24 são apresentados os resultados da ANOVA para a interação entre os parâmetros. O teste identificou diferença estatística significativa ao nível de 5% de significância para os parâmetros pop, λ e para a interação pop * λ .

Tabela 24 – Teste de honestidade de Tukey ao nível de 5% de significância para DWWO no cenário RGM

Fatores	Soma quadrados	Graus de liberdade	Média quadrados	Estatística F	Significância
pop	245,531	3	81,844	9,251	0,000
λ	285,464	4	71,366	8,067	0,000
hmax	2,846	2	1,423	0,161	0,851
ω	9,956	2	4,978	0,563	0,570
pop * λ	241,283	12	20,107	2,273	0,007
pop * hmax	5,929	6	0,988	0,112	0,995
pop * ω	38,842	6	6,474	0,732	0,624
λ * hmax	6,036	8	0,755	0,085	1,000
λ * ω	29,226	8	3,653	0,413	0,914
hmax * ω	4,301	4	1,075	0,122	0,975

Fonte: Autor, 2019.

Afim de avaliar a diferença estatística dentro de cada fator avaliado, um teste de Tukey foi conduzido para os parâmetros pop e λ . O teste identificou a existência de dois subgrupos para o parâmetro pop, o primeiro contendo pop = {25, 35, 15} com médias do desvio relativo correspondentes a 2,385, 2,401 e 2,409, respectivamente e o último contendo pop = {5} com média aproximada de 2,463. Para o parâmetro λ foram identificados também dois subgrupos contendo $\lambda = \{(1, 2), (1, 4), (1, 3)\}$ com respectivas médias iguais a 2,38, 2,392 e 2,394 e $\lambda = \{(2, 3), (2, 4)\}$ com médias correspondente a 2,452 e 2,453.

Tabela 25 - ANOVA para interação entre pop * λ

pop	(I) λ	(J) λ	Diferença da média (I-J)	Erro Padrão	Significância
5	(1 e 2)	(1 e 3)	-0,029	0,035	0,995
		(1 e 4)	-0,037	0,035	0,965
		(2 e 3)	-0,186*	0,035	0,000
		(2 e 4)	-0,178*	0,035	0,000
	(1 e 3)	(1 e 2)	0,029	0,035	0,995
		(1 e 4)	-0,008	0,035	1,000
		(2 e 3)	-0,157*	0,035	0,000
		(2 e 4)	-0,149*	0,035	0,000
	(1 e 4)	(1 e 2)	0,037	0,035	0,965
		(1 e 3)	0,008	0,035	1,000
		(2 e 3)	-0,149*	0,035	0,000
		(2 e 4)	-0,140*	0,035	0,001
	(2 e 3)	(1 e 2)	0,186*	0,035	0,000
		(1 e 3)	0,157*	0,035	0,000
		(1 e 4)	0,149*	0,035	0,000
		(2 e 4)	0,008	0,035	1,000
	(2 e 4)	(1 e 2)	0,178*	0,035	0,000
		(1 e 3)	0,149*	0,035	0,000
		(1 e 4)	0,140*	0,035	0,001
		(2 e 3)	-0,008	0,035	1,000

Fonte: Autor, 2019.

Por meio da aplicação da ANOVA para a interação entre λ * pop, foi identificado diferença estatística entre médias das faixas de λ dentro do subconjunto pop = 5. Os resultados obtidos pela ANOVA podem ser vistos na Tabela 25.

Figura 53 - Desempenho do algoritmo no cenário RGM para interação entre pop*λ

Fonte: Autor, 2019.

Pela interação entre $pop * \lambda$, vista na Figura 53, é observado que a menor média do desvio relativo são obtidas para $pop = 25$ e $\lambda = (1, 2)$. Para o cenário RGM, os seguintes parâmetros foram adotados: $pop = 25$, $\lambda = (1, 2)$, $\omega = 40$ e $hmax = 20$.

4.3.3 Análise dos resultados das metaheurísticas para minimização de C_{max} para problemas de pequeno porte

Nesta seção são apresentados os resultados do desempenho das heurísticas em comparação às soluções dadas MILP proposto na Seção 3. Foram comparados também o efeito da busca local proposta para o reposicionamento das atividades de manutenção preventiva (MP) em uma dada sequência. Por ser computacionalmente custosa, a busca local foi aplicada apenas para a solução final dada pela metaheurística. Neste sentido, a solução final gerada pelas metaheurísticas são rotuladas como DWWO Padrão, IG_RIS Padrão e IG_IJ Padrão, e a aplicação da busca local às soluções finais dadas pelas metaheurísticas são rotuladas como DWWO Busca Local, IG_RIS Busca Local e IG_IJ Busca Local, respectivamente. Todas as configurações encontradas como as melhores nas subseções que trataram da parametrização das

metaheurísticas são utilizadas nesta Seção. Foi estabelecido como critério de parada o tempo computacional de $n \times m \times l \times 30$ milissegundos. A heurística PFT-NEH(5)st foi utilizada para gerar a restrição de custo (MP_{cost}) e a sequência inicial das metaheurísticas propostas.

O IBM CPLEX encontrou soluções dentro de um limite de 3.600 segundos por instância para até $n = 8$, $m = 5$ e $l = 2$, no cenário YS. Para instâncias em que $n = 6$ e 7 e $l > 2$, o MILP encontrou solução para a maior parte dos 40 problemas por instância, com exceção de $n = 7$, $m = 5$ e $l = 4$, onde foi obtida solução de 4 problemas. No total, o IBM CPLEX conseguiu solucionar 2410 problemas no cenário YS. Para o cenário RGM, o IBM CPLEX encontrou soluções para até $n = 8$, $m = 5$ e $l = 3$. Por se tratar de problemas mais fáceis de serem resolvidos em comparação com os do cenário YS, o CPLEX conseguiu encontrar soluções para problemas mais complexos, como por exemplo, a instância $n = 7$, $m = 5$ e $l = 4$, onde conseguiu encontrar solução de 24 problemas e $n = 8$, $m = 4$ e $l = 4$, solucionando 34 problemas. Dos 2.560 problemas, o MILP retornou a solução de 2.498. Os resultados do desempenho das heurísticas pela combinação $n \times m$ podem ser vistas na Tabela 26.

Tabela 26 - Desempenho das metaheurísticas em comparação ao MILP no cenário YS (melhores resultados destacados em negrito)

n	m	Modelo	DWWO Padrão	DWWO Busca Local	IG_RIS Padrão	IG_RIS Busca Local	IG_IJ Padrão	IG_IJ Busca Local
5	2	0	5,31	2,57	4,62	2,11	4,23	1,76
	3	0	16,15	12,12	15,54	11,52	15,55	11,78
	4	0	21,67	16,46	21,50	16,48	21,63	17,08
	5	0	23,40	18,81	23,34	18,75	23,01	18,52
6	2	0	6,19	3,43	5,04	2,30	5,04	2,18
	3	0	18,55	14,67	17,38	13,39	17,12	13,73
	4	0	24,21	19,87	23,00	18,73	22,80	18,56
	5	0	28,92	23,78	25,75	21,44	25,63	21,20
7	2	0	6,91	4,40	5,12	3,12	5,37	3,21
	3	0	21,55	18,11	18,29	15,00	18,55	15,02
	4	0	26,87	22,17	23,49	19,78	23,37	19,73
	5	0	29,42	25,61	25,42	22,53	25,72	22,67
8	2	0	8,55	6,31	6,32	4,35	6,04	3,91
	3	0	22,42	19,08	18,33	15,50	18,44	15,42
	4	0	28,21	24,42	23,64	21,03	23,70	20,73
	5	0	29,67	27,37	25,43	23,15	25,72	23,05
Média Geral		0	19,30	15,58	17,18	13,82	17,14	13,78

Fonte: Autor, 2019.

Pelos resultados mostrados na Tabela 26, pode-se observar que IG_IJ e IG_RIS obtiveram os melhores resultados que DWWO. Em especial, a aplicação da busca local obteve uma melhoria significativa em relação a versão padrão das metaheurísticas. Na média geral, a aplicação da busca local à solução final apresentou melhoria de 3,72% para DWWO e 3,36% para IG_RIS e IG_IJ.

Tabela 27 - Desempenho das metaheurísticas em comparação ao MILP no cenário RGM (melhores resultados destacados em negrito)

<i>n</i>	<i>m</i>	Modelo	DWWO	DWWO	IG_RIS	IG_RIS Busca	IG_IJ	IG_IJ Busca
			Padrão	Busca Local	Padrão	Local	Padrão	Local
5	2	0	2,68	2,64	2,45	1,32	2,37	1,20
	3	0	11,64	11,32	11,39	9,68	11,13	9,28
	4	0	18,48	17,98	17,75	15,11	17,78	15,24
	5	0	21,48	20,80	21,05	17,60	20,83	17,40
6	2	0	4,60	3,72	3,70	1,87	3,75	1,79
	3	0	16,92	15,45	15,56	12,71	15,64	12,98
	4	0	21,53	20,03	19,87	16,44	19,74	16,08
	5	0	26,01	24,15	24,92	21,04	24,81	20,92
7	2	0	5,96	4,93	4,21	2,78	4,04	2,68
	3	0	17,47	15,68	15,05	12,88	15,30	12,92
	4	0	23,24	21,09	21,02	17,81	21,18	18,01
	5	0	29,49	27,09	26,28	23,24	26,72	23,47
8	2	0	6,94	5,67	4,39	2,92	4,91	3,27
	3	0	19,91	17,81	16,60	14,31	16,84	14,40
	4	0	26,91	24,29	22,87	19,99	22,84	20,31
	5	0	28,47	26,52	23,89	21,61	24,65	22,04
Média		0	17,34	15,95	15,48	13,00	15,56	13,03
Geral								

Fonte: Autor, 2019.

Na Tabela 27 pode-se constatar que, assim como no cenário YS, o desempenho das heurísticas propostas tendem a piorar significativamente a medida que o número de tarefas e máquinas aumenta. Pelos valores apresentados, pode-se constatar que as buscas locais também afetaram significativamente a qualidade de solução de todas as metaheurísticas. Para DWWO, a diferença média entre os desvios relativos da versão Padrão e da Busca Local é de aproximadamente 1,39%. Já as diferenças entre IG_RIS Padrão e IG_RIS Busca Local e IG_IJ Padrão e IG_IJ Busca Local são de 2,48% e 2,53% , respectivamente.

Figura 54- Gráfico de Médias ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Pelos gráficos a) e b) apresentados na Figura 54 pode ser constatado que IG_IJ, IG_RIS e DWWO Busca Local apresentaram os melhores desempenhos e suas médias diferem significativamente de suas versões Padrões. Pelo fato de apresentar as melhores soluções para o problema, o ponto correspondente ao modelo foi omitido com o propósito de melhor visualizar a distância entre as metaheurísticas.. Pode-se notar também que IG_IJ e IG_RIS Busca Local obtiveram o melhor desempenho e suas médias podem ser consideradas estatisticamente diferentes de DWWO Busca Local, ao nível de 5% de significância. Já em RGM, DWWO Busca Local tende a produzir soluções piores que IG_RIS Padrão e IG_Padrão. Porém, pode-se visualizar que não há diferença significativa estatística ao nível de 5% de significância entre os três métodos avaliados. Pode-se também observar que para o cenário RGM, IG_RIS Busca Local tende a obter uma média levemente menor que IG_IJ Busca Local, o contrário do que ocorre em YS, porém, sem diferença estatística significativa.

O teste de honestidade de Tukey apresentado na Tabela 28 identificou cinco subconjuntos homogêneos distintos de médias. Pode-se constatar que a aplicação da busca local às soluções geradas por IG_IJ e IG_RIS apresentaram os melhores desempenhos.

Tabela 28 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário YS

Métodos	N	Subconjuntos homogêneos com $\alpha = 0,05$				
		1	2	3	4	5
Modelo	2.410	0,000				
IG_IJ Busca Local	2.410		13,780			
IG_RIS Busca Local	2.410		13,819			
DWWO Busca Local	2.410			15,578		
IG_IJ Padrão	2.410				17,143	
IG_RIS Padrão	2.410				17,178	
DWWO Padrão	2.410					19,296
Significância		1,000	1,000	1,000	1,000	1,000

Fonte: Autor, 2019.

Já para o cenário RGM, o teste de honestidade de Tukey apresentado na Tabela 29 identificou 4 subconjuntos homogêneos para a amostra analisada. Assim como em YS, em RGM, IG_RIS Busca Local e IG_IJ Busca Local apresentaram o melhor desempenho.

Tabela 29 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário RGM

Métodos	N	Subconjuntos para $\alpha = 0.05$			
		1	2	3	4
Modelo	2.498	0,000			
IG_RIS Busca Local	2.498		12,996		
IG_IJ Busca Local	2.498		13,031		
IG_RIS Padrão	2.498			15,477	
IG_IJ Padrão	2.498			15,560	
DWWO Busca Local	2.498			15,948	
DWWO Padrão	2.498				17,342
Significância		1,000	1,000	0,850	1,000

Fonte: Autor, 2019.

Em relação ao número de tarefas n , pode-se constatar pelos gráficos a) e b) apresentados na Figura 55 que, em geral, a medida que n aumenta, todas as metaheurísticas tendem a obter soluções mais afastadas das geradas pelo modelo.

Figura 55 - Desempenho das metaheurísticas em relação à n para problemas de pequeno porte nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

DWWO Padrão obtém os resultados mais distantes da solução ótima. Já a aplicação da busca local à solução gerada por DWWO apresenta diferenças expressivas na comparação entre cenários. Os gráficos a) e b) da Figura 55 sugerem que o desempenho de DWWO tende a piorar em comparação com IG_IJ e IG_RIS em situações em que os intervalos de MP são mais espaçados, que é o caso do cenário RGM. Dentre as metaheurísticas avaliadas, IG_IJ Busca Local e IG_RIS Busca Local tendem a obter os melhores desempenhos com o aumento de n . Em todos os casos de n , a busca local obteve melhoria significativa em relação a solução final Padrão dada pelas metaheurísticas.

Em relação ao número de máquinas pode-se também constatar que a medida que m aumenta, o desempenho das metaheurísticas tendem a piorar nos dois cenários. No cenário YS, IG_RIS Busca Local e IG_IJ Busca Local, assim como também IG_IJ Padrão e IG_RIS Padrão, apresentam soluções muito similares em todas as instâncias de m . Já em RGM, pode-se notar que o desempenho de DWWO Busca Local, IG_IJ Padrão e IG_RIS Padrão são similares para todos os valores assumidos de m . Pelos gráficos a) e b) dispostos na Figura 56 pode-se observar que o fator m interfere de modo similar o desempenho das metaheurísticas em ambos os cenários avaliados.

Figura 56 - Desempenho das metaheurísticas em relação à m para problemas de pequeno porte nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Já em relação a duração dos tempos de *setup*, observa-se nos gráficos a) e b) da Figura 57 um comportamento diferente do desempenho das metaheurísticas em comparação com n e m . No cenário YS os resultados mostram que com o aumento da taxa de *setup*, o desvio relativo das metaheurísticas tendem a diminuir. Já no cenário RGM pode-se constatar que o desempenho das heurísticas tende a melhorar a medida que a duração do *setup* aumenta, exceto quando assume valores de até 125% dos tempos de processamento.

Figura 57 - Desempenho das metaheurísticas em relação ao *setup* para problemas de pequeno porte nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Pelos gráficos a) e b) da Figura 57, pode-se perceber de maneira mais evidente o efeito do uso das buscas locais em relação aos níveis de manutenção (l). Em geral, é observado diferenças expressivas na diferença entre as soluções padrões dadas pelas metaheurísticas e a

aplicação da busca local, especialmente quando $l = 4$, onde pode-se notar a maior diferença entre os desvios relativos da versão Padrão das metaheurísticas e as respectivas Buscas Locais. Algumas pequenas diferenças podem ser vistas na comparação entre cenários. Enquanto que em YS, a aplicação das buscas locais tende a suavizar a curva do desvio médio relativo para todas as metaheurísticas, no cenário RGM, a busca local aplicada a DWWO apresenta semelhanças de comportamento com a sua versão Padrão para todos os valores de l .

Figura 58 - Desempenho das metaheurísticas em relação à l para problemas de pequeno porte nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Pela análise conjunta dos dois cenários avaliados, pode-se constatar que para problemas de pequeno porte IG_RIS e IG_IJ são os métodos que obtiveram os melhores resultados. Pode-se visualizar também que o uso da busca local apresentou melhorias significativas para a solução das metaheurísticas. Portanto, recomenda-se o uso de IG_RIS Busca Local e IG_IJ Busca Local para ambos os cenários YS e RGM.

4.3.4 Análise dos resultados para problemas de grande porte

Nesta seção são apresentados os resultados do desempenho das metaheurísticas em relação ao banco de instâncias de problemas de médio e grande porte. A análise dos resultados foi análoga à realizada para os problemas de pequeno porte, ou seja, além da avaliação das soluções finais dadas pelas heurísticas, foi comparado também o impacto da aplicação da busca local proposta para o reposicionamento das atividades de manutenção preventiva (MP) em uma

dada sequência. Logo, a solução final gerada pelas metaheurísticas é rotulada como DWWO Padrão, IG_RIS Padrão e IG_IJ Padrão, e a aplicação da busca local às soluções finais dadas pelas metaheurísticas é rotulada como DWWO Busca Local, IG_RIS Busca Local e IG_IJ Busca Local, respectivamente. Foi estabelecido como critério de parada o tempo computacional de $n \times m \times l \times 30$ milissegundos para cada problema. A restrição de custo, a solução inicial de IG_IJ e IG_RIS e o primeiro indivíduo de DWWO foram gerados pela heurística PFT-NEH(5)st. Os resultados dos desvios relativos médios agrupados por $n \times m$ para o cenário YS podem ser vistos na Tabela 30.

Tabela 30 - Desempenho das metaheurísticas para problemas de médio e grande porte agrupados por $n \times m$ no cenário YS

n	m	DWWO	DWWO Busca	IG_IJ	IG_IJ Busca	IG_RIS	IG_RIS
		Padrão	Local	Padrão	Local	Padrão	Busca Local
20	5	4,642	3,578	3,633	1,876	3,824	1,960
	10	5,062	4,030	3,345	1,440	4,016	2,084
	15	4,555	3,443	3,222	1,431	3,492	1,689
	20	4,656	3,685	3,011	1,232	3,414	1,626
50	5	5,139	4,641	2,007	1,185	2,199	1,466
	10	4,735	4,092	2,265	1,234	2,354	1,378
	15	4,716	4,176	2,309	1,302	2,343	1,387
	20	4,368	3,821	2,261	1,280	2,244	1,220
100	5	2,520	2,323	2,073	1,643	2,533	2,153
	10	1,339	1,101	2,133	1,538	2,352	1,791
	15	0,984	0,727	1,977	1,428	2,319	1,683
	20	0,771	0,506	2,378	1,663	2,033	1,385
200	5	0,300	0,200	2,484	2,207	2,063	1,815
	10	0,325	0,188	1,610	1,211	1,455	1,098
	15	0,232	0,111	1,524	1,149	1,457	1,085
	20	0,227	0,109	1,169	0,812	1,175	0,796
500	5	0,039	0,010	0,679	0,573	0,682	0,575
	10	0,072	0,022	0,351	0,227	0,351	0,227
	15	0,082	0,023	0,341	0,191	0,341	0,191
	20	0,075	0,025	0,360	0,211	0,360	0,211
Média Geral		2,242	1,841	1,957	1,192	2,050	1,291

Fonte: Autor, 2019.

Como pode ser constatado na Tabela 30, IG_IJ e IG_RIS obtiveram as menores médias gerais. Entretanto, pode-se observar que, para o critério de parada estabelecido, DWWO obtém

os menores desvios relativos para problemas a partir de $n = 100$ e $m = 10$. Na Tabela 31 podem ser visualizados os desvios relativos médios agrupados por $n \times m$ para o cenário RGM.

Tabela 31 - Desempenho das heurísticas para problemas de médio e grande porte agrupados por $n \times m$ no cenário RGM

n	m	DWWO	DWWO Busca	IG_IJ	IG_IJ Busca	IG_RIS	IG_RIS Busca
		Padrão	Local	Padrão	Local	Padrão	Local
20	5	4,136	2,337	5,041	2,517	4,835	2,342
	10	4,603	2,416	5,055	2,103	5,067	2,139
	15	4,768	2,818	3,941	1,348	4,563	2,035
	20	4,575	2,827	3,833	1,172	4,385	1,858
50	5	4,503	3,341	2,968	1,487	2,577	0,986
	10	4,921	3,314	3,314	1,294	3,118	1,054
	15	5,019	3,371	3,185	1,076	3,055	1,096
	20	5,527	4,042	2,992	1,030	2,925	1,016
100	5	2,935	2,016	2,283	1,202	2,259	1,250
	10	2,808	1,311	3,298	1,412	3,351	1,507
	15	2,999	1,447	3,638	1,505	3,224	1,273
	20	2,600	1,037	3,220	1,016	3,301	1,055
200	5	1,162	0,502	2,607	1,680	2,657	1,709
	10	1,580	0,394	2,633	0,951	2,640	0,862
	15	1,713	0,309	2,728	0,738	2,749	0,754
	20	1,718	0,325	2,638	0,569	2,623	0,583
500	5	0,493	0,040	1,427	0,608	1,442	0,618
	10	0,872	0,070	1,442	0,254	1,475	0,259
	15	0,920	0,072	1,495	0,255	1,495	0,254
	20	1,065	0,100	1,556	0,224	1,561	0,216
Média Geral		2,946	1,604	2,965	1,122	2,965	1,143

Fonte: Autor, 2019.

Algumas diferenças de desempenho das metaheurísticas no cenário RGM podem ser pontuadas em comparação com o cenário YS. Na média geral do desvio relativo, IG_IJ Busca Local e IG_RIS Busca Local obtiveram as melhores médias gerais do desvio relativo. Entretanto, pode-se constatar que DWWO torna-se competitivo a partir de $n = 100$.

Tabela 32 - Teste de honestidade de Tukey para problemas de médio e grande porte no cenário YS

Metaheurísticas	N	Subconjuntos para $\alpha = 0.05$			
		1	2	3	4
IG_IJ Busca Local	3.200	1,191			
IG_RIS Busca Local	3.200	1,291			
DWWO Busca Local	3.200		1,840		
IG_IJ Padrão	3.200		1,956	1,956	
IG_RIS Padrão	3.200			2,050	
DWWO Padrão	3.200				2,241
Significância		0,561	0,382	0,625	1,000

Fonte: Autor, 2019.

O teste de honestidade de Tukey apresentado na Tabela 32 identificou 4 subconjuntos homogêneos. Neste caso, IG_IJ Busca Local e IG_RIS Busca Local obtiveram as menores médias do desvio relativo. Observa-se que IG_IJ Padrão é equiparável a DWWO Busca Local, uma vez que ambas estão no subconjunto 2. O mesmo pode ser dito de IG_IJ Padrão e IG_RIS Padrão que estão inclusas no subconjunto 3.

Tabela 33 - Teste de honestidade de Tukey para problemas de médio e grande porte no cenário RGM

Metaheurística	N	Subconjuntos para $\alpha = 0.05$		
		1	2	3
IG_IJ Busca Local	3.200	1,122		
IG_RIS Busca Local	3.200	1,143		
DWWO Busca Local	3.200		1,604	
DWWO Padrão	3.200			2,945
IG_IJ Padrão	3.200			2,964
IG_RIS Padrão	3.200			2,965
Significância		0,999	1,000	0,999

Fonte: Autor, 2019.

O teste honestidade de Tukey apresentado na Tabela 33 apresentou resultados diferentes em comparação com o cenário YS. No cenário RGM, foram identificados três subconjuntos homogêneos, sendo que, DWWO Busca Local pertence a um subconjunto e DWWO Padrão, IG_IJ Padrão e IG_RIS Padrão estão em outro subgrupo. Pode-se também constatar que apesar de não apresentarem diferença estatística significativa ao nível de 5% de significância, DWWO apresenta um desvio relativo médio levemente menor que IG_RIS e IG_IJ.

Figura 59 - Gráfico de médias do desvio relativo ao nível de 95% de confiança para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Pelos gráficos a) e b) apresentados na Figura 59, pode-se observar resultados similares aos resultados gerados pelo teste de honestidade de Tukey. Neste sentido, o emprego da busca local para a solução final dada pelas metaheurísticas afeta significativamente a qualidade de solução. Em especial, constata-se que, com a aplicação da busca local, os resultados de IG_IJ Busca Local, IG_RIS Busca Local e DWWO Busca Local apresentaram diferença estatística significativa ao nível de 5% de significância em comparação com suas versões Padrão em ambos os cenários. As diferenças de desempenho das metaheurísticas na versão Padrão ficam mais evidentes no cenário YS. Entretanto, percebe-se que DWWO Busca Local apresenta um desempenho melhor em RGM se comparado ao cenário YS.

Figura 60 - Desempenho das metaheurísticas em relação à n para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Em relação ao número de tarefas (n), pode-se constatar pelos gráficos a) e b) da Figura 60 que o desempenho das metaheurísticas tendem a se aproximar a medida que n aumenta. É notável que a distância entre as soluções dadas pelas versões com Busca Local das metaheurísticas tendem a se aproximar das soluções dadas pelas respectivas versões Padrão com o aumento de n . A partir de 100 tarefas, DWWO Padrão e DWWO Busca Local tendem a obter os menores desvios relativos. No cenário RGM, pode-se observar diferenças expressivas entre os desempenhos DWWO Padrão, IG_RIS Padrão e IG_IJ Padrão e as soluções dadas pelas respectivas correspondentes com aplicação busca local.

Figura 61 - Desempenho das metaheurísticas em relação à m para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Pelo número de máquinas m , conforme visto nos gráficos a) e b) da Figura 61, fica evidente que IG_RIS e IG_IJ tendem a gerar soluções superiores em comparação a DWWO. De modo geral, todas as metaheurísticas tendem a obter menores desvios relativos com o aumento de m . Na média por valor de m , IG_RIS e IG_IJ Busca Local obtiveram os melhores resultados, com distância significativa em comparação a suas versões Padrão e DWWO. Já DWWO Busca Local, diferente do que ocorre no gráfico de n , apresenta desvios relativos médios menores que IG_IJ Padrão e IG_RIS Padrão, obtendo também distância expressiva em comparação a DWWO Padrão. No cenário RGM, pode ser constatado diferenças significativas no comportamento dos desvios relativos entre DWWO Padrão e DWWO Busca Local, IG_IJ Padrão e IG_IJ Busca Local e IG_RIS Padrão e IG_RIS Busca Local. Em especial, é observável que o uso da busca local tende a diminuir o desvio relativo das soluções finais dadas por IG_IJ e IG_RIS com o aumento de m .

Figura 62 - Desempenho das metaheurísticas em relação aos tempos de *setup* para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Em relação a duração do *setup*, pode-se constatar que, de modo geral, o desvio relativo médio das heurísticas tende a piorar com o aumento da duração. Por se tratarem da mesma sequência gerada, a busca local tende a gerar resultados com comportamento similar a da versão Padrão, porém, com desvios relativos médios menores. Pelos gráficos a) e b) da Figura 62 é observado que IG_RIS é o método que obteve o menor ponto médio quando a duração do *setup* corresponde a até 50% dos tempos de processamento. Diferentemente do que ocorre no cenário YS, IG_RIS obtém o melhor resultado quando a duração do *setup* corresponde a até 125% dos tempos de processamento.

Figura 63 - Desempenho das metaheurísticas em relação à *l* para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Já em relação aos níveis de manutenção preventiva l , pode-se observar pela Figura 63 que a busca local afeta significativamente a qualidade da solução com o aumento de l nos dois cenários YS (a) e RGM (b) avaliados. Ao contrário do que ocorre com DWWO Busca Local, os desvios relativos médios de IG_IJ Busca Local e IG_RIS Busca Local tendem a diminuir, obtendo as menores médias quando $l = 5$. Já em RGM, os dois referidos métodos tendem a obter resultados similares com o aumento de l . Estes resultados impactam de forma expressiva a qualidade da solução dada por IG_IJ Padrão e IG_RIS Padrão, uma vez que a solução final dada por ambas tende a piorar com o aumento de l .

Portanto, pelos resultados apresentados anteriormente, observa-se que IG_IJ obteve o melhor desempenho geral nos dois cenários YS e RGM, o que sugere sua aplicação para os dois cenários avaliados.

4.4 CONSIDERAÇÕES FINAIS DA SEÇÃO 4

Nesta Seção foram apresentados métodos de solução para a minimização da duração total da programação sujeita ao custo total de manutenção preventiva (MP) para o problema *flow shop* com bloqueio, *setup* dependente da sequência e (MP) baseada em níveis. Por se tratar de um problema condicional, foi necessário primeiro, estabelecer o valor do custo total de MP e uma solução inicial para o emprego das metaheurísticas adaptadas ao problema.

Com base na revisão de literatura apresentada na Seção 2, foram adaptadas 14 heurísticas construtivas para o problema de minimização do custo total de MP (*total_cost*). Adicionalmente 8 métodos heurísticos construtivos, considerados versões alternativas de algumas heurísticas adaptadas, foram também propostas ao problema. No total, 22 métodos heurísticos construtivos foram avaliados. Pode-se constatar experimentalmente que PFT-NEH(5) obteve o melhor desempenho no *trade-off* entre eficiência e eficácia. Entretanto, ressalta-se que os gráficos apresentados nas Figuras 41 e 42 fornecem os métodos que compõem a curva de pareto, podendo, portanto, serem utilizados para determinar o método mais apropriado para o tomador de decisão, dado os recursos de tempo disponíveis.

Das metaheurísticas avaliadas, dado o critério de parada adotado, pode-se concluir que IG_IJ Busca Local é o método que obteve o melhor desempenho tanto para ambas as instâncias de pequeno, médio e grande portes dos dois cenários avaliados YS e RGM. Entretanto, pode-se observar também que IG_RIS é competitiva, além de ser de mais fácil implementação em

relação a IG_IJ. Logo, as duas metaheurísticas supracitadas podem ser sugeridas para solucionar o problema adotado nesta Seção.

Para pesquisas futuras, sugere-se a avaliação do desempenho das metaheurísticas com outros intervalos de tempo para o critério de parada. Pode ser observado por experimentações computacionais que intervalos menores que $n \times m \times l \times 30$ milissegundos não afetam significativamente o desempenho das heurísticas. Portanto, sugere-se como critério de parada diferentes faixas de tempo acima do que foi estabelecido nesta seção. Esta análise pode auxiliar na avaliação de comportamento das heurísticas ao longo do tempo. Uma outra sugestão para pesquisas futuras é a adaptação de metaheurísticas propostas para outras funções objetivo, como tempo total de fluxo e atraso total, para o problema estudado.

5 MÉTODOS DE SOLUÇÃO PARA O PROBLEMA DE MINIMIZAÇÃO DE TEMPO TOTAL DE FLUXO SUJEITO AO CUSTO TOTAL DE MANUTENÇÃO PREVENTIVA

Nesta Seção são apresentadas heurísticas e metaheurísticas utilizadas para o problema $F_m | block, NM_j^t, s_{j,k} | (\sum_i C_i | total_cost = C)$. Levou-se em consideração duas vias de solução: Método exato com aplicação do modelo de programação linear inteira mista (MILP) e proposição de heurísticas e metaheurísticas para o problema. De forma análoga ao que foi proposto na Seção 4, as heurísticas construtivas propostas para os problemas $F_m | block | \sum_i C_i$, identificadas durante a revisão de literatura da Seção 2, foram adaptadas e com o objetivo de minimizar prioritariamente o *total_cost*. Posteriormente, a melhor heurística avaliada é utilizada como geradora de soluções iniciais para as metaheurísticas da literatura adaptadas ao problema. Experimentações computacionais foram conduzidas no intuito de testar o desempenho dos métodos e avaliar suas diferenças de desempenho nos dois cenários diferentes de MP gerados.

5.1 HEURÍSTICAS CONSTRUTIVAS PARA MINIMIZAÇÃO DO TEMPO TOTAL DE FLUXO DADO O CUSTO TOTAL DE MANUTENÇÃO PREVENTIVA

Nesta seção são apresentadas heurísticas construtivas para minimização do custo total de manutenção (*total_cost*) e como solução inicial para as metaheurísticas propostas para o problema. Os seguintes métodos foram adaptados para solução do problema: MME-A e MME-B de HAN et al. (2013), HPF1, HPF2, NHPF1 e NHPF2 de Ribas e Companys (2015), tPF-NEH e tPF-NEH(x) de Tasgetiren et al. (2016). Outras heurísticas propostas para o problema como *Beam Search* de Fernandez-Viagas et al. (2016) e NEH-MK de Moslehi e Khorasanian (2013) não foram incluídas nas análises por exigirem tempo computacional muito maior que em relação às outras heurísticas. As heurísticas MME-A e MME-B foram descritas na Seção 4.1.2, logo, serão omitidas neste capítulo.

5.1.1 HPF1, HPF2, NHPF1 e NHPF2

Propostas por Ribas e Companys (2015), HPF1 e HPF2 são inspiradas na heurística PF e levam em consideração a minimização do tempo total de fluxo. Para a seleção da primeira tarefa, HPF1 considera a tarefa que obtém a menor data de término. A próxima tarefa a ser inserida na próxima posição da sequência parcial é a que minimiza o custo total de manutenção preventiva (MP). Em casos de empate, a próxima tarefa da sequência parcial é selecionada de acordo com o seguinte critério:

$$ind(i,k) = \alpha \cdot \left(\sum_{j=1}^m \left(D_{i,j} - D_{[k],j} - p_{i,j} - s_{j,[k],i} - \sum_{t=1}^l MA_{[k],i,j}^t \right) \right) + (1 - \alpha) \cdot (D_{i,m} - D_{[k],m} - s_{j,[k],i} - \sum_{t=1}^l MA_{[k],i,j}^t) \quad (68)$$

Onde:

$ind(i,k)$ – valor da medida ao inserir a tarefa i após a tarefa que ocupa a posição k da sequência parcial

$D_{i,j}$ – Instante de partida da tarefa i na máquina j na posição $k + 1$ da sequência parcial;

$D_{[k],j}$ – Instante de partida da tarefa que ocupa a posição k da sequência parcial na máquina j ;

$s_{j,[k],i}$ – Tempo de *setup* na máquina j entre a tarefa que ocupa a posição k da sequência e a tarefa candidata i ;

α – Parâmetro que assume valores dentro da faixa $[0, 1]$.

Portanto, como critério de desempate, a tarefa i que minimiza o índice $ind(i,k)$ é removida do conjunto de tarefas não programadas e inserida na posição $k + 1$ da sequência parcial. Este processo é repetido até que todas as tarefas sejam inseridas na sequência parcial. Em NHPF1 o mecanismo de inserção *NEH_Insertion_Scheme* (Figura 9) é aplicado à sequência gerada por HPF1. O pseudocódigo das heurísticas HPF1/NHPF1 podem ser vistos na Figura 64.

```

//HPF1/NHPF1
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$  e  $\pi'' \leftarrow \emptyset$ 
//HPF1
Remova a tarefa  $\pi(k)$  da sequência  $\pi$  que gera o menor  $C_{max}$  e insira em  $\pi'(1)$ 
Para  $i \leftarrow 2$  até  $n$  Faça
    Remova a tarefa  $\pi(k)$  de  $\pi$  que minimiza o custo total de MP. Em caso de empate, escolha a tarefa  $\pi(k)$ 
    que minimiza  $ind(i,k)$  e insira em  $\pi'(i)$ 
FimPara
// NHPF1
Aplique NEH_Insertion_Scheme( $\pi'$ ,  $\pi''$ )
Retorne  $\pi''$ 
End //HPF1/NHPF1

```

Figura 64 - Pseudocódigo das heurísticas HPF1/NHPF1

Fonte: Autor, 2019.

Em HPF2 é considerado também a minimização do espaço de ociosidade (*front delay*) que as máquinas devem esperar para processar a primeira tarefa. Neste sentido, Ribas e Companys (2015) obtém o valor do *front delay* utilizando relações geométricas. Entretanto, conforme ressaltado pelos autores supracitados, a escolha da primeira tarefa depende também da data de conclusão da tarefa e o *front delay* resultante. Assim, os autores estabeleceram um índice bicritério $R(i)$, que faz a ponderação dos dois termos. De modo a adicionar o tempo de *setup* da primeira tarefa, a medida $R(i)$ pode ser calculada de acordo com o procedimento descrito na Figura 65.


```

//R(i)
 $\pi \leftarrow$  Sequência de tarefas
//HPF2
Para  $i \leftarrow 1$  até  $n$  Faça
   $front\_delay(i) \leftarrow 0$ 
   $t, b \leftarrow 1$ 
   $C_{i,1} \leftarrow p_{i,1} + s_{1,i,i}$ 
  Para  $j \leftarrow 2$  até  $m$  Faça
    Se  $C_{i,j-1} > s_{j,i,i}$  Então
       $C_{i,j} \leftarrow C_{i,j-1} + p_{i,j}$ 
       $front\_delay(i) \leftarrow front\_delay(i) + p_{i,j} \cdot (m - j - 1)$ 
    Senão
       $C_{i,j} \leftarrow s_{j,i,i} + p_{i,j}$ ,  $front\_delay(i) \leftarrow 0$ 
       $t \leftarrow j$ ;  $b \leftarrow 1$ 
    FimSenão
  FimPara  $j$ 
  Para  $j \leftarrow t+1$  até  $m$  Faça
     $b \leftarrow b + 1$ 
  FimPara  $j$ 
  Se  $b = 1$  Então
     $R(i) = \lambda \cdot (C_{i,m} + p_{i,m}) + (1 - \lambda) \cdot C_{i,m}$ 
  Senão
     $R(i) = \lambda \cdot [(front\_delay(i)) \cdot 2 / ((b-1)) + s_{m,i,i}] + (1 - \lambda) \cdot C_{i,m}$ 
  FimSenão
FimPara  $i$ 
Retorne  $\pi$ 
Fim //R(i)

```

Figura 65 - Pseudocódigo do procedimento para obtenção do índice $R(i)$

Fonte: Autor, 2019.

O valor de λ também é parametrizável, assumindo valores dentro da faixa $[0,1]$.

```

//HPF2/NHPF2
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$  e  $\pi'' \leftarrow \emptyset$ 
//HPF2
Remova a tarefa  $\pi(k)$  da sequência  $\pi$  que minimiza  $R(i)$ . Em caso de empate, escolha a tarefa que minimiza  $C_{1,1}$ 
Para  $i \leftarrow 2$  até  $n$  faça
  Remova a tarefa  $\pi(k)$  de  $\pi$  que minimiza o custo total de MP. Em caso de empate, escolha a tarefa  $\pi(k)$  que minimiza  $ind(i,k)$  e insira em  $\pi'(i)$ 
FimPara
// NHPF2
Aplique NEH_Insertion_Scheme( $\pi'$ ,  $\pi''$ )
Retorne  $\pi''$ 
Fim //HPF2/NHPF2

```

Figura 66 - Pseudocódigo das heurísticas HPF2/NHPF2

Fonte: Autor, 2019.

A tarefa que apresenta o menor índice $R(i)$ é removida do conjunto de tarefas não programadas e inserida na primeira posição da sequência parcial. A partir da segunda posição, seleciona-se a tarefa que apresenta o menor custo total de manutenção preventiva ($total_cost$). Em caso de empates, o critério estabelecido pela Eq. (68) é utilizado para escolha da próxima tarefa. O pseudocódigo das heurísticas HPF2/NHPF2 podem ser vistas na Figura 66.

5.1.2 tPF-NEH(x)

Proposta por Tasgetiren et al. (2016), tPF-NEH(x) segue mecanismos de funcionamento semelhantes a PFT-NEH(x) e PF-NEH(x). Como forma de inicialização do algoritmo, tPF-NEH(x) necessita de uma lista inicial (LI) utilizada para selecionar a primeira tarefa a ser inserida na sequência parcial. Tasgetiren et al. (2016) utilizam o índice $R(i)$, desenvolvido por Ribas e Companys (2015) para geração de LI. A partir da segunda posição, a tarefa que minimiza o $total_cost$ é escolhido. Em caso de empate, o critério de seleção proposto pelos autores é utilizado.

Duas medidas são adotadas para escolha da tarefa a ser inserida na última posição da sequência parcial. A primeira medida leva em conta a soma total dos tempos de ociosidade e bloqueio que a tarefa i causa ao ser inserida após a tarefa que ocupa a posição k da sequência, ou seja, $\delta_{[k],i} = \sum_{j=1}^m (D_{i,j} - D_{[k],j} - p_{i,j} - s_{j,[k],i} - \sum_{t=1}^l MA_{[k],i,j}^t)$. A segunda medida (SD_i) leva em conta a soma ponderada dos instantes de partida das tarefas em todas as máquinas:

$$SD_i = \sum_{j=1}^{m-1} \left(\frac{mR_{i-1,j+1}}{j + i(m-j)/(n-2)} \right) \quad (69)$$

Onde:

$R_{i,j}$ – Instante de Início da tarefa i na máquina j

Com os valores de $\delta_{[k],i}$ e SD_i calculados e conhecidos, o seguinte critério de desempate CM_i é estabelecido para escolha da tarefa que vai ocupar a última posição da sequência parcial:

$$CM_i = (1 - \mu)\delta_{[k],i} + \mu \cdot SD_i \quad (70)$$

Onde μ é um parâmetro que assume valores dentro da faixa $[0, 1]$.

O processo é repetido até que todas as tarefas sejam programadas. O pseudocódigo de tPF pode ser visto na Figura 67.

```

//tPF
 $\pi \leftarrow$  Sequência de tarefas,  $\pi' \leftarrow \emptyset$ 
Remova a tarefa  $\pi(k)$  da sequência  $\pi$  que minimiza  $R(i)$ . Em caso de empate, escolha a tarefa que minimiza  $C_{1,1}$ 
Para  $i \leftarrow 2$  até  $n$  Faça
    Remova a tarefa  $\pi(k)$  de  $\pi$  que minimiza o custo total de MP. Em caso de empate, escolha a tarefa  $\pi(k)$  que minimiza  $CM_i = (1 - \mu)\delta_{[k],i} + \mu \cdot SD_i$  e insira em  $\pi'(i)$ 
FimPara
Retorne  $\pi'$ 
Fim //tPF

```

Figura 67 - Pseudocódigo de tPF

Fonte: Autor, 2019.

O pseudocódigo da heurística tPF-NEH(x) pode ser visto na Figura 68. Após a aplicação de tPF, a sequência π' é guardada como solução atual. Na segunda fase da heurística, o mecanismo *NEH_Insertion_Scheme*(λ) é aplicado à sequência gerada por tPF. Durante esta fase, é escolhida a sequência parcial que minimiza o *total_cost*. Em caso de empates é escolhida a sequência que minimiza o tempo total de fluxo. Após a aplicação do mecanismo *NEH_Insertion_Scheme*(λ), uma nova sequência π'' é gerada. Se esta sequência apresenta um *total_cost* menor que a sequência atual π' , π'' se torna a sequência atual π' . Por fim, se a nova sequência atual π' apresenta custo total de MP menor que a melhor sequência π_{best} encontrada até o momento, então, π_{best} recebe a sequência π' . O algoritmo retorna para a fase tPF, selecionando agora a próxima tarefa da LI para ocupar a primeira posição da sequência. O processo é repetido para x tarefas da LI.

```

// tPF-NEH(x)
 $\pi \leftarrow$  Sequência de tarefas,  $\pi'$ ,  $\pi''$  e  $\pi_{best} \leftarrow \emptyset$ 
 $bestcost, bestcmax \leftarrow$  Número muito grande
Ordene as tarefas em  $\pi$  ordem crescente de  $R(i)$ , conforme procedimento da Figura 65.
For  $p \leftarrow 1$  to  $x$  do
     $\pi'(1) \leftarrow \pi(p)$ 
     $\pi' \leftarrow tPF-NEH(x)$ 
     $Cost \leftarrow total\_cost(\pi')$ ,  $Cmax \leftarrow C_{max}(\pi')$ 
    //Inserção NEH( $\lambda$ )
    Aplique NEH( $\lambda$ )_Insertion_Scheme( $\pi'$ ,  $\pi''$ )

    If  $Cost < total\_cost(\pi'')$ 
         $Cost \leftarrow total\_cost(\pi'')$ ,  $Cmax \leftarrow C_{max}(\pi'')$ 
         $\pi' \leftarrow \pi''$ 
    Else
        If  $Cost = total\_cost(\pi'')$  and  $Cmax > C_{max}(\pi'')$  then
             $Cost \leftarrow total\_cost(\pi'')$ ,  $Cmax \leftarrow C_{max}(\pi'')$ 
             $\pi' \leftarrow \pi''$ 
        endif
    endelse

    If  $bestcost < Cost$ 
         $bestcost \leftarrow Cost$ ,  $bestcmax \leftarrow C_{max}(\pi')$ 
         $\pi_{best} \leftarrow \pi'$ 
    Else
        If  $bestcost = Cost$  and  $bestcmax > Cmax$  then
             $bestcost \leftarrow Cost$ ,  $bestcmax \leftarrow Cmax$ 
             $\pi_{best} \leftarrow \pi'$ 
        endif
    endelse
endfor
Retorne  $\pi_{best}$ 
End // tPF-NEH(x)

```

Figura 68 – Pseudocódigo das heurísticas tPF-NEH(x)

Fonte: Autor, 2019.

5.2 METAHEURÍSTICAS PARA MINIMIZAÇÃO DO TEMPO TOTAL DE FLUXO

Baseado na revisão de literatura apresentada na Seção 2, três recentes metaheurísticas foram adaptadas ao problema estudado: GRASP(HPF2) de Ribas e Companys (2015), DABC_RCT de Ribas et al. (2015) e VBIH de Tasgetiren et al. (2016). Ambos os métodos foram provados como os mais eficazes e mais recentes para o problema $F_m|block|\sum_i C_i$. Nas seções posteriores são descritos os métodos.

5.2.1 Greedy Randomized Adaptive Search Procedure (GRASP)

A metaheurística GRASP, originalmente proposta por Feo e Resende (1989), consiste de iterações feitas de construções sucessivas a partir de uma solução gulosa aleatorizada e seguida de uma busca local que tenta melhorar a solução até que um determinado critério de parada seja atendido. Ribas e Companys (2015) propõem quatro versões de GRASP, utilizando as heurísticas HPF1, HPF2, NHPF1 e NHPF2 como gerador de soluções para a primeira parte da metaheurística. Pelos resultados apresentados, os autores sugerem o uso de GRASP(HPF2) ou GRASP(NHPF2), que obtiveram desempenhos muito similares nos estudos experimentais realizados. Pela eficiência do algoritmo, adotou-se a versão GRASP(HPF2) para ser adaptada ao problema proposto. O pseudocódigo do algoritmo pode ser visto na Figura 69.

```

//GRASP(HPF2)
Best_TFT  $\leftarrow \infty$ 
 $\lambda \leftarrow \text{Valor}(0, 1)$ 
Repita
   $\mu \leftarrow \mu_{min} + (\mu_{max} - \mu_{min}) \cdot \text{aleatório}(0, 1)$ 
   $\pi_2 \leftarrow \text{HPF2}(\lambda, \mu)$ 
   $\pi_2 \leftarrow \text{local\_search}(\pi_2)$ 
  Se  $\text{total\_cost}(\pi_2) \leq MP_{cost}$  e  $\sum_i C_i(\pi_2) < \text{Best\_TFT}$  então
     $\pi \leftarrow \pi_2$ 
     $\text{Best\_TFT} \leftarrow \sum_i C_i(\pi_2)$ 
Até(Critério_de_Parada)
Fim //GRASP(HPF2)

```

Figura 69 - Pseudocódigo de GRASP(HPF2)

Fonte: Autor, 2019.

No algoritmo apresentado na Figura 69, o valor de μ é aleatoriamente sorteado de uma distribuição uniforme entre μ_{min} e μ_{max} a cada iteração e o valor de λ é fixo, sendo estabelecido por meio de experimentações computacionais (Seção 5.3). Na fase de busca local, os autores adotam um algoritmo VNS com duas estruturas de vizinhança: inserção (LS2) e troca entre pares de tarefas (LS1). O pseudocódigo da heurística pode ser visto na Figura 70.

Na busca local LS2, os vizinhos são gerados permutando a tarefa da sequência em todas as posições seguintes a ela. Se o melhor vizinho encontrado é melhor que a solução incumbente, esta se torna a nova solução incumbente. O processo continua até que todas as tarefas sejam testadas.

Na busca LS2, os vizinhos são gerados inserindo uma tarefa da sequência em todas as posições possíveis. Se o melhor vizinho encontrado é melhor que a solução incumbente, esta se torna a nova solução incumbente e o processo continua até que todas as tarefas sejam testadas. Para evitar que a vizinhança seja explorada na mesma ordem, as tarefas são selecionadas aleatoriamente.

```

// VNS
 $\pi_1 \leftarrow \pi$ 
 $tempF \leftarrow \sum_i C_i(\pi_1)$ 
 $contador \leftarrow 0$ 
Se (aleatório < 0,5) então
    LS = 0
Senão
    LS = 1
FimSe
Enquanto(flag = true) faça
     $contador \leftarrow contador + 1$ 
    Se LS = 0 então
         $\pi_2 \leftarrow LS1(\pi_1)$ 
    Senão
         $\pi_2 \leftarrow LS2(\pi_1)$ 
    FimSe
    Se ( $cost\_MP(\pi_2) \leq LBcost\_MP$  e  $\sum_i C_i(\pi_2) < tempF$ ) ou ( $contador = 1$ ) então
        LS  $\leftarrow 1 - LS$ 
         $\pi_1 \leftarrow \pi_2$ 
         $tempF \leftarrow \sum_i C_i(\pi_2)$ 
    FimSe
FimEnquanto
Retorne  $\pi$ 
Fim // busca local VNS

```

Figura 70 – Pseudocódigo do algoritmo VNS

Fonte: Autor, 2019.

Conforme pode ser visto na Figura 70, a primeira estrutura de vizinhança a ser explorada é selecionada aleatoriamente com probabilidade de 50%. Após explorar soluções vizinhas da solução incumbente, a solução atual é comparada com a solução incumbente. Se houver melhoria, a solução atual substitui a solução incumbente e a busca continua na outra vizinhança. Esse processo continua até que a solução incumbente não obtenha mais melhorias. Por fim, após passar pela fase de busca local, a nova solução encontrada é comparada com a melhor solução. Se esta solução apresentar um *total_cost* menor ou igual ao da melhor solução e seu tempo total de fluxo é menor que o da melhor, então a nova solução torna-se a melhor solução. O processo é repetido então até que o critério de parada seja atendido.

5.2.2 Discrete Artificial Bee Colony (DABC)

O algoritmo *Artificial Bee Colony* é uma metaheurística de inteligência de enxame bio-inspirada no comportamento de forrageamento das abelhas (KARABOGA; BASTURK, 2008). O algoritmo possui essencialmente três componentes: as fontes de alimento, que correspondem ao conjunto de soluções; as abelhas campeiras que estão associadas com uma fonte de alimento específica a ser explorada e as abelhas não-campeiras. As abelhas não-campeiras consistem em: abelhas observadoras, que esperam no ninho e estabelecem uma fonte de alimento por meio das informações compartilhadas pelas abelhas campeiras; e abelhas exploradoras, que procuram novas fontes de alimento pelas áreas que rodeiam a colmeia (KARABOGA; BASTURK, 2008, AKAY; KARABOGA, 2012). A partir do funcionamento básico descrito anteriormente, Ribas et al. (2015) propõem a combinação de diversas estratégias de busca, intensificação e diversificação nas fases de geração das fontes de alimento e nas fases em que as abelhas campeiras, observadoras e exploradoras são utilizadas. Por meio de um *design* de experimento, os autores encontram as estratégias mais eficazes para cada fase da metaheurística. Baseando-se nesta combinação, a metaheurística DABC_RCT proposta por Ribas et al. (2015) foi aplicada ao problema proposto. As subseções a seguir descrevem o funcionamento da metaheurística.

5.2.2.1 Inicialização

O algoritmo inicia-se com a geração de N soluções iniciais. Ribas et al. (2015) propõem o uso da heurística HPF2 para geração das soluções iniciais, sendo a primeira solução dada pela melhor combinação encontrada entre λ e μ , e o restante das soluções fixando-se o valor de λ e escolhendo aleatoriamente o valor de μ na faixa $[\mu_{min}, \mu_{max}]$. Os valores de μ_{min} e μ_{max} seguiram as mesmas configurações de Ribas e Companys (2015). Uma pequena adaptação é feita para a primeira fonte de alimento, que será gerada pela heurística construtiva que apresentar o melhor desempenho para minimização do *total_cost*. Os parâmetros μ_{min} e μ_{max} foram atribuídos de acordo com estudos experimentais apresentados na Seção 5.3.2.1.

5.2.2.2 Abelhas Campeiras

Nesta fase, as abelhas campeiras são enviadas para as fontes de alimentos no intuito de avaliar seus arredores. Os autores empregam duas fontes de alimento durante esta fase: a melhor solução encontrada e outra escolhida aleatoriamente. Para melhorar a exploração e ser capaz de acessar boas fontes de alimentos, os autores adotam um operador denominado *Three Neighborhood Operators* (TNO). Este procedimento foi proposto inicialmente por Croce et al. (1996) e consiste em aplicar três operadores para duas soluções selecionadas. Esses operadores são definidos como seguem:

- PI (Troca entre pares de tarefas): Dada uma sequência π e duas posições, $k1$ e $k2$, troque as duas tarefas de posições;
- ESR (Extração e Reinscrição em Posição Posterior): Dada uma sequência π e duas posições $k1$ e $k2$, com $k2$ posterior a $k1$, extraia a tarefa que ocupa a posição $k2$ e insira na posição $k1$;
- ESR (Extração e Reinscrição em Posição Anterior): Dada uma sequência π e duas posições $k1$ e $k2$, onde $k1$ vem antes de $k2$, extraia a tarefa que ocupa a posição $k1$ e reinsira na posição $k2$.

O procedimento TNO inicia-se selecionando aleatoriamente $k1$ e $k2$ ($k1 < k2$). Então, os três operadores são aplicados a sequência selecionada π e a melhor solução das três é escolhida e passa pela busca local VNS, descrita na Figura 70, com a exceção de que LS1 corresponde a estrutura de vizinhança de troca entre pares de tarefas e LS2 corresponde a inserção. O pseudocódigo de TNO pode ser visto na Figura 71.


```

// TNO
Para  $j \leftarrow 1$  até  $t$  Faça
  Selecione duas posições  $k1$  e  $k2$  aleatoriamente de 1 a  $n$ 
   $i1 \leftarrow \pi(k1)$ ;  $i2 \leftarrow \pi(k2)$ 
   $\pi1 \leftarrow PI(\pi, k1, k2)$ 
   $\pi2 \leftarrow PI(\pi, i2, k1)$ 
   $\pi3 \leftarrow PI(\pi, i1, k2)$ 
   $\pi' \leftarrow$  Sequência que gerou o menor custo e como critério de desempate, o menor tempo total de fluxo
  Se  $total\_cost(\pi') \leq MP_{cost}$  e  $\sum_i C_i(\pi') < \sum_i C_i(\pi)$  Então
     $\pi \leftarrow \pi'$ 
  FimSe
FimPara
 $\pi'' \leftarrow VNS(\pi)$  //Figura 70
Se  $total\_cost(\pi'') \leq MP_{cost}$  e  $\sum_i C_i(\pi'') < \sum_i C_i(\pi)$  Então
   $\pi \leftarrow \pi''$ 
FimSe
Retorne  $\pi$ 
Fim // TNO

```

Figura 71 – Pseudocódigo do procedimento TNO

Fonte: Autor, 2019.

5.2.2.3 Abelhas Observadoras

As abelhas observadoras procuram por novas fontes de alimentos para serem exploradas. Elas esperam no ninho e estabelecem a fonte de alimento por meio das informações compartilhadas pelas abelhas campeiras. Nesta fase é aplicada a estratégia de *path relinking* utilizando como sequência referencia a melhor solução encontrada até então e outra sequência aleatoriamente escolhida. A estratégia *path relinking* já foi descrita na Seção 4.2.2.2 e seu pseudocódigo pode ser visto na Figura 24.

5.2.2.4 Abelhas exploradoras

As abelhas exploradoras procuram por novas fontes de alimento. Nesta fase, uma nova solução é criada de acordo com a estratégia seguida na fase de geração das fontes de alimento, ou seja, uma nova solução é gerada por $HPF2(\lambda, \mu)$. A pior solução da população é então substituída se o *total_cost* da nova solução for menor ou igual e se o $\sum_i C_i$ da nova solução for

menor que o da pior solução. Por fim, o pseudocódigo da metaheurística DABC pode ser visto na Figura 72.

```
// DABC
Inicialização
Primeiro indivíduo da fonte de alimentos ← Heurística Construtiva
Restante das fontes de alimentos ← HPF2( $\lambda, \mu$ )
 $\pi_{\text{best}} \leftarrow$  Melhor indivíduo da população
 $\pi_{\text{worst}} \leftarrow$  Pior indivíduo da população

Enquanto(Critério_de_Parada) Faça
// Abelhas campeiras
Para  $\pi_{\text{best}}$  e outro indivíduo da população escolhido da população faça:
    Para  $j \leftarrow 1$  até  $t$  faça
        Aplique o procedimento TNO (Figura 71)
    fimPara
FimPara

//Abelha Observadora
Selecione um indivíduo aleatoriamente da população e aplique path relinking em direção a  $\pi_{\text{best}}$ 

//Abelha exploradora
 $\pi_{\text{new}} \leftarrow$  HPF2( $\lambda, \mu$ )
Se  $\text{total\_cost}(\pi_{\text{new}}) \leq \text{total\_cost}(\pi_{\text{worst}})$  e  $\sum_i C_i(\pi_{\text{worst}}) > \sum_i C_i(\pi_{\text{new}})$  então
     $\pi_{\text{worst}} \leftarrow \pi_{\text{new}}$ 
FimSe
FimEnquanto
Retorne  $\pi$ 
Fim // DABC
```

Figura 72 – Pseudocódigo da metaheurística DABC

Fonte: Autor, 2019.

5.2.3 Variable Block Insertion Heuristic (VBIH)

Proposta por Tasgetiren et al. (2016), a metaheurística *Variable Block Insertion Heuristic* (VBIH) utiliza a idéia de inserção de blocos de tarefas como forma de diversificar a vizinhança de busca. Os autores denotam dois tamanhos de bloco $bS_{\min}$, tidos como tamanho de bloco mínimo e $bS_{\max}$, definido como o tamanho máximo do bloco. Adicionalmente, os autores também definem um tamanho máximo de movimento de inserção, denotado como $mS_{\max}$. Basicamente, VBIH remove um bloco de tarefas com tamanho bS da sequência atual, iniciando com $bS_{\min} = 1$, e então, um número máximo $mS_{\max}$ de movimentos é realizado aleatoriamente na sequência parcial, denotado como procedimento de inserção de blocos (bIM). A melhor sequência gerada por bIM é sujeita ao algoritmo *Variable Local Search* (VLS), que é baseada

nas estruturas de vizinhança de inserção e troca entre pares de tarefas. Se a melhor sequência obtida é melhor que a sequência atual, esta se torna a sequência atual. Enquanto há melhoria, o tamanho do bloco continua o mesmo, ou seja, $bS \leftarrow bS$. Caso contrário, o tamanho do bloco é incrementado em 1, ou seja, $bS \leftarrow bS + 1$, e um critério de aceitação baseado em *Simulated Annealing* é usado para avaliar se a solução inferior é adotada ou não, com o objetivo de escapar de ótimos locais. O processo é repetido até que o tamanho do bloco atinja o máximo de blocos, ou seja, $bS \leq bS_{max}$. O pseudocódigo geral de VBIH pode ser visto na Figura 73.

```
// VBIH
 $\pi, \pi', \pi_{best} \leftarrow$  Heurística Construtiva
 $MP_{cost} \leftarrow total\_cost(\pi)$ 
Enquanto(Critério de Parada) Faça
     $bS = bS_{min} = 1$ 
    Enquanto ( $bS \leq bS_{max}$ ) Faça
         $\pi_1 \leftarrow bIM(\pi, bS, mS_{max})$ 
         $\pi_2 \leftarrow VLS(\pi_1)$ 
        Se  $total\_cost(\pi_2) \leq MP_{cost}$  e  $\sum_i C_i(\pi_2) < \sum_i C_i(\pi)$  Então
             $\pi \leftarrow \pi_2$ 
             $bS \leftarrow bS$ 
            Se  $total\_cost(\pi_2) \leq MP_{cost}$  e  $\sum_i C_i(\pi_2) < \sum_i C_i(\pi_{best})$  Então
                 $\pi_{best} \leftarrow \pi' \leftarrow \pi_2$ 
            FimSe
        Senão
             $bS \leftarrow bS + 1$ 
            Se  $total\_cost(\pi_2) \leq MP_{cost}$  e  $\sum_i C_i(\pi_2) > \sum_i C_i(\pi)$  Então
                Se ( $r < \exp\{-(\sum_i C_i(\pi) - \sum_i C_i(\pi_2))/T\}$ ) Então
                     $\pi \leftarrow \pi_2$ 
                FimSe
            FimSe
        FimSenão
    FimEnquanto
Retorne  $\pi_{best}$ 
End // VBIH
```

Figura 73 – Pseudocódigo da metaheurística VBIH

Fonte: Autor, 2019.

O procedimento de movimento de inserção de blocos $bIM(\pi, bS, mS_{max})$ é a função principal em VBIH. O procedimento remove um bloco de tarefas de tamanho bS da sequência atual π , denotada como π^b e remove este bloco de tarefas de π , gerando $\pi^P \leftarrow \pi^P - \{\pi^b\}$. Então, o bloco π^b é inserido na sequência parcial π^P em posições aleatórias mS_{max} vezes. Por fim, a melhor sequência das mS_{max} geradas é escolhida. O pseudocódigo do procedimento $bIM(\pi, bS, mS_{max})$ é descrito na Figura 74.

```

// bIM
 $\pi_1, \pi^P \leftarrow \pi$ 
 $MP_{cost} \leftarrow total\_cost(\pi)$ 
tempF = Número muito grande
 $\pi^b \leftarrow$  Bloco de tarefas de tamanho  $bS$  removidas de  $\pi_1$ 
 $\pi^P \leftarrow \pi^P - \{\pi^b\}$ 
Para  $i \leftarrow 1$  até  $mSmax$  Faça
  pt  $\leftarrow$  Escolha uma posição em  $\pi^P$  aleatoriamente
   $\pi_2 \leftarrow$  Insira o bloco  $\pi^b$  na posição pt de  $\pi^P$ 
  Se  $total\_cost(\pi_2) \leq MP_{cost}$  e  $\sum_i C_i(\pi_2) < tempF$  Então
     $\pi_1 \leftarrow \pi_2$ 
    tempF  $\leftarrow \sum_i C_i(\pi_2)$ 
  FimSe
FimPara
Retorne  $\pi_1$ 
Fim // bIM

```

Figura 74 – Pseudocódigo do procedimento de inserção bIM

Fonte: Autor, 2019.

Após a aplicação de bIM(), conforme descrito na Figura 73, o algoritmo *Variable Local Search* (VLS) é aplicado à sequência π_1 . O pseudocódigo do VLS proposto pode ser visto na Figura 75.

```

// VLS
contador  $\leftarrow 0$ 
Se ( $r < 0,5$ ) então
  LS = 0
Senão
  LS = 1
FimSe
Enquanto (flag = true) faça
  contador  $\leftarrow$  contador + 1
  tempF  $\leftarrow \sum_i C_i(\pi_1)$ 
  Se LS = 0 então
     $\pi_2 \leftarrow LS1(\pi_1)$ 
  Senão
     $\pi_2 \leftarrow LS2(\pi_1)$ 
  FimSe
  Se ( $total\_cost(\pi_2) \leq MP_{cost}$  e  $\sum_i C_i(\pi_2) < tempF$ ) ou (contador = 1) então
    LS  $\leftarrow 1 - LS$ 
     $\pi_1 \leftarrow \pi_2$ 
    tempF  $\leftarrow \sum_i C_i(\pi_2)$ 
  FimSe
FimEnquanto
Retorne  $\pi$ 
Fim // VLS

```

Figura 75 – Pseudocódigo do algoritmo VLS

Fonte: Autor, 2019.

Os procedimentos de LS1 e LS2 consistem em estruturas de vizinhança de inserção e troca entre pares de tarefas, respectivamente. A busca local utilizada em LS1 é correspondente à busca local *Reference Insertion Search*, descrito na Figura 21.a). Já a busca local utilizada em LS2, denominada *Referenced Permutation Search* pode ser vista na Figura 25, com a exceção de que π^R é dado por π' .

Após a aplicação da busca local, caso a nova solução encontrada seja pior que a solução incumbente, um critério de aceitação baseado em *Simulated Annealing* com uma temperatura constante para avaliar se a nova solução é aceita como solução incumbente. Como já descrito em *Iterated Greedy*, a temperatura T é dada pela Eq. (65).

5.3 RESULTADOS COMPUTACIONAIS

Nesta seção são apresentados resultados experimentais do modelo matemático aplicado ao objetivo de desempenho adotado e os desempenhos das heurísticas construtivas e das metaheurísticas. Foram utilizados todas as medidas e bancos de instâncias de problemas descritas na Seção 3.6.

5.3.1 Avaliação computacional das heurísticas construtivas

Todas as heurísticas construtivas foram testadas nos dois bancos de problemas de médio e de grande porte (20 a 500 tarefas). Como já mencionado, os métodos testados nesta seção tiveram como prioridade a minimização do custo total de manutenção (*total_cost*), seguido pela minimização do tempo total de fluxo. Inicialmente, experimentos computacionais foram conduzidos com as bases referidas, no intuito de calibrar os parâmetros exigidos pelas heurísticas MM-A, MM-B, HPF1, HPF2 e tPF-NEH. Para tPF-NEH(x) o número de tarefas λ foi estabelecido seguindo as configurações de Tasgetiren et al. (2016), ou seja, $\lambda = 15$ tarefas. Os resultados obtidos pela experimentação computacional de HPF1 para os cenários YS (a) e RGM (b) podem ser visualizados na Figura 76.

Figura 76 - Calibração do parâmetro μ da heurística HPF1 nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Os parâmetros que apresentaram o melhor desempenho, ou seja, $\mu = 0$ para o cenário YS e $\mu = 0,05$ para o cenário RGM foram adotados como solução da heurística HPF1. Na Figura 77, apresenta-se a calibração da heurística MM-B para os dois cenários YS (a) e RGM (b). Conforme os dados apresentados pela Figura 77, tPF obteve o melhor desempenho quando α assume o valor 0,95 para o cenário YS e 1 para o cenário RGM.

Figura 77 - Calibração do parâmetro μ da heurística tPF nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Figura 78 - Calibração do parâmetro α da heurística MMA nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Pelos gráficos a) e b) apresentados na Figura 78, pode ser constatado que para os dois cenários avaliados, MM-A apresenta o melhor desempenho quando $\alpha = 0,5$, sugerindo que a heurística obtém melhores valores quando o peso ponderado do índice de avaliação é igual para os dois termos avaliados. Analogamente, conforme pode ser visto nos gráficos a) e b) da Figura 79, MM-B também atinge seu melhor desempenho em $\alpha = 0,5$.

Figura 79 - Calibração do parâmetro α da heurística MMA nos cenários a) YS e b) RGM

Fonte: Autor, 2019.

Para a heurística HPF2, os valores de μ e λ foram calibrados levando-se em conta a melhor combinação entre eles, ou seja, a configuração de $\mu \times \lambda$ que obteve o melhor desempenho. Os resultados de todas as combinações podem ser vistos nas Tabelas 55 e 56, localizadas no apêndice A. Portanto, para o cenário YS, $\lambda = 0,7$ e $\mu = 0,1$ e para o cenário RGM, $\lambda = 0,3$ e $\mu = 0,3$

Tabela 34 - Desempenho das heurísticas construtivas para minimização do custo total de manutenção preventiva e tempo total de fluxo no cenário YS (Continua...)

n	m	HPF1	HPF2	NHPF1	NHPF2	tPF-NEH	tPF-NEH(1)	tPF-NEH(2)	tPF-NEH(5)	MM-A	MM-B	MME-A	MME-B
20	5	11,5	11,9	4,90	5,05	4,86	4,59	2,74	0,89	11,01	11,06	5,24	5,04
	10	7,34	7,74	3,27	3,38	3,45	3,39	1,93	0,46	6,80	7,03	3,18	3,33
	15	6,35	6,14	2,76	2,96	2,47	2,18	1,35	0,31	5,94	6,04	2,60	2,56
	20	5,42	5,31	2,58	2,71	2,49	2,38	1,45	0,27	5,07	5,14	2,43	2,33
50	5	5,84	6,11	2,54	2,61	2,42	2,68	1,79	0,65	4,67	4,52	2,38	2,36
	10	3,41	3,53	2,11	1,84	1,96	2,16	1,30	0,51	3,38	3,42	1,71	1,93
	15	3,01	2,93	1,52	1,68	1,67	1,73	1,03	0,41	2,93	2,94	1,60	1,57
	20	2,55	2,52	1,51	1,48	1,55	1,57	0,96	0,34	2,42	2,48	1,57	1,59
100	5	4,03	3,89	2,40	2,36	2,25	1,72	1,18	0,45	2,73	2,67	2,16	2,21
	10	2,30	2,30	1,95	1,90	1,90	1,58	0,96	0,32	1,93	1,98	1,86	1,77
	15	1,81	1,79	1,68	1,65	1,65	1,16	0,71	0,23	1,58	1,58	1,58	1,60
	20	1,39	1,39	1,34	1,46	1,45	0,95	0,56	0,14	1,31	1,36	1,56	1,56

Tabela 34 - Desempenho das heurísticas construtivas para minimização do custo total de manutenção preventiva e tempo total de fluxo no cenário YS (Conclusão.)

n	m	HPF 1	HPF 2	NHPF 1	NHPF 2	tPF- NEH	tPF- NEH(1)	tPF- NEH(2)	tPF- NEH(5)	MM- A	MM- B	MME- A	MME- B
200	5	2,76	2,78	2,61	2,66	2,62	1,27	0,73	0,24	1,57	1,62	2,63	2,50
	10	1,50	1,57	2,06	2,07	2,19	0,92	0,54	0,10	1,20	1,25	2,17	2,13
	15	1,08	1,16	1,92	1,93	2,01	0,82	0,43	0,07	1,00	1,03	2,00	1,99
	20	0,95	0,97	1,88	1,93	1,87	0,68	0,37	0,09	0,84	0,81	1,81	1,84
500	5	2,22	2,29	3,28	3,21	3,18	1,02	0,64	0,32	0,61	0,64	3,20	3,22
	10	1,02	0,98	2,57	2,62	2,59	0,56	0,37	0,09	0,56	0,54	2,63	2,64
	15	0,72	0,64	2,42	2,46	2,48	0,49	0,28	0,08	0,58	0,56	2,45	2,44
	20	0,56	0,51	2,38	2,38	2,35	0,41	0,23	0,03	0,51	0,50	2,38	2,39
Média Geral		3,29	3,32	2,38	2,42	2,37	1,61	0,98	0,30	2,83	2,86	2,36	2,35

Fonte: Autor, 2019.

Tabela 35- Desempenho das heurísticas construtivas para minimização do custo total de manutenção preventiva e tempo total de fluxo no cenário RGM

n	m	HPF 1	HPF 2	NHPF 1	NHPF 2	tPF- NEH	tPF- NEH(1)	tPF- NEH(2)	tPF- NEH(5)	MM- A	MM- B	MME- A	MME- B
20	5	13,26	13,48	6,15	5,96	5,90	5,28	3,06	0,89	11,03	10,76	5,69	5,59
	10	7,91	8,27	3,83	3,71	3,68	3,27	1,87	0,44	7,09	7,11	3,95	3,82
	15	6,75	7,00	3,21	3,26	3,07	2,84	1,68	0,38	6,26	6,28	2,87	2,95
	20	5,60	5,55	2,89	2,78	2,62	2,43	1,44	0,48	5,24	5,17	2,58	2,68
50	5	6,92	6,97	3,42	3,14	3,09	2,76	1,74	0,76	4,57	4,90	2,95	2,99
	10	4,13	4,52	2,36	2,50	2,11	2,30	1,43	0,63	3,59	3,60	2,32	2,34
	15	3,14	3,07	1,88	1,97	2,02	1,89	1,16	0,35	2,83	2,87	1,75	1,80
	20	2,65	2,67	1,77	1,77	1,81	1,61	0,99	0,27	2,39	2,40	1,83	1,67
100	5	4,86	5,35	2,96	2,98	2,85	2,47	1,35	0,52	2,49	2,52	2,70	2,86
	10	3,11	2,95	2,06	2,16	2,16	1,73	1,21	0,50	2,03	2,09	2,08	2,03
	15	2,05	2,03	1,81	1,92	1,95	1,30	0,85	0,24	1,80	1,81	1,91	1,96
	20	1,57	1,85	1,89	1,84	1,80	1,27	0,74	0,27	1,63	1,61	1,85	1,83
200	5	4,03	4,65	3,22	3,33	3,13	1,56	0,93	0,33	1,19	1,16	3,30	3,27
	10	2,00	1,99	2,09	2,25	2,23	1,05	0,68	0,28	1,09	1,10	2,20	2,14
	15	1,47	1,40	1,96	1,97	1,94	0,86	0,55	0,20	1,02	1,05	1,99	1,97
	20	1,04	1,09	1,83	1,88	1,96	0,74	0,43	0,12	0,92	0,84	1,92	1,94
500	5	4,40	5,18	4,51	4,49	4,32	1,49	1,15	0,73	0,44	0,47	4,49	4,43
	10	1,72	1,84	2,45	2,35	2,36	0,74	0,45	0,18	0,50	0,51	2,38	2,40
	15	0,98	0,99	2,06	2,07	2,06	0,53	0,32	0,13	0,54	0,50	2,02	2,07
	20	0,72	0,66	2,13	2,13	2,07	0,47	0,32	0,10	0,48	0,52	2,11	2,10
Média Geral		3,92	4,08	2,73	2,72	2,66	1,83	1,12	0,39	2,86	2,86	2,64	2,64

Fonte: Autor, 2019.

Assim como foi constatado no cenário YS, a heurística tPF-NEH(x), em especial, a versão tPF-NEH(5), obteve o melhor desempenho para o cenário RGM. Constata-se também similaridades no desempenho das heurísticas restantes. A partir de 200 tarefas e 10 máquinas,

observa-se que MME-A, MME-B, NHPF1 e NHPF2 tendem a gerar soluções piores que MM-A, MM-B, HPF1 e HPF2, respectivamente. Pelas Tabelas 34 e 35 pode ser constatado que tPF-NEH(5) obteve os melhores resultados em todas as instâncias de problemas avaliadas. Pode-se também observar que o mecanismo de inserção adotado por tPF-NEH(x) tende a se desempenhar melhor que tPF-NEH, versão da heurística que considera a inserção pelo mecanismo de NEH (Figura 9) de todas as tarefas da sequência gerada por tPF. De fato, a partir de 100 tarefas, a versão tPF-NEH(1) obtém médias menores que em relação à todas as outras heurísticas avaliadas, exceto tPF-NEH(2) e tPF-NEH(5). Na Figura 80 são mostradas as diferenças estatísticas ao nível de 5% de significância entre as heurísticas avaliadas.

Figura 80 - Gráfico de médias ao nível de 5% de significância das heurísticas para minimização do custo total de manutenção preventiva e tempo total do fluxo nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

No cenário YS, é possível observar que, na média geral, tPF-NEH(5) obtém o melhor desempenho entre todas as heurísticas, seguidas de suas versões tPF-NEH(2) e tPF-NEH(1). As três versões apresentam diferença estatística significativa em comparação com todas as outras heurísticas avaliadas. Nota-se também que heurísticas que adotaram o mecanismo de inserção de NEH apresentaram desempenhos muito similares, assim como MM-A e MM-B e HPF1 e HPF2. O gráfico b), mostrado na Figura 80, aponta aproximações entre as heurísticas que adotam o mecanismo NEH e as heurísticas MM-A e MM-B. Diferentemente do que ocorre no cenário YS, em RGM, MM-A e MM-B não apresentam diferença estatística significativa com NHPF1, NHPF2, tPF-NEH, MME-A e MME-B.

Tabela 36 - Teste de honestidade de Tukey ao nível de 5% de significância para as heurísticas construtivas no cenário YS

Heurísticas	N	Suconjuntos para $\alpha = 0.05$					
		1	2	3	4	5	6
tPF-NEH(5)	3.200	0,300					
tPF-NEH(2)	3.200		0,978				
tPF-NEH(1)	3.200			1,613			
MME-B	3.200				2,350		
MME-A	3.200				2,356		
tPF-NEH	3.200				2,369		
NHPF1	3.200				2,384		
NHPF2	3.200				2,417		
MM-A	3.200					2,833	
MM-B	3.200					2,858	
HPF1	3.200						3,287
HPF2	3.200						3,323
Significância		1,000	1,000	1,000	,996	1,000	1,000

Fonte: Autor, 2019.

Tabela 37 - Teste de honestidade de Tukey ao nível de 5% de significância para as heurísticas construtivas no cenário RGM

Heurísticas	N	Subconjuntos para $\alpha = 0.05$				
		1	2	3	4	5
tPF-NEH(5)	3.200	0,389				
tPF-NEH(2)	3.200		1,117			
tPF-NEH(1)	3.200			1,831		
MME-B	3.200				2,643	
MME-A	3.200				2,644	
tPF-NEH	3.200				2,657	
NHPF2	3.200				2,724	
NHPF1	3.200				2,725	
MM-A	3.200				2,856	
MM-B	3.200				2,863	
HPF1	3.200					3,916
HPF2	3.200					4,076
Significância		1,000	1,000	1,000	0,075	0,499

Fonte: Autor, 2019.

Pela Tabela 36, observa-se que o teste de honestidade de Tukey identificou 6 subconjuntos de médias homogêneas no cenário YS. De fato, todas as versões de tPF-NEH(x) obtiveram média do desvio relativo menor que as outras heurísticas avaliadas com diferença

estatística significativa. Pode-se constatar também que MME-A, MME-B, NHPF1 e NHPF2 melhoram expressivamente as soluções geradas por MMA, MMB, HPF1 e HPF2, respectivamente. Como explicitado na Tabela 37, o teste de Tukey identificou a existência de 5 subconjuntos homogêneos no cenário RGM, um a menos do que foi identificado em YS. Outros subgrupos permanecem inalterados, ou seja, tPF-NEH(5), tPF-NEH(2) e tPF-NEH(1) apresentam diferenças estatísticas significativas entre eles e outras heurísticas e HPF1 e HPF2 pertencem ao mesmo subconjunto.

Figura 81 - Desempenho das heurísticas construtivas em relação à n para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Pelo número de tarefas (n), pode ser constatado pelos gráficos a) e b) da Figura 81 que, de modo geral, os desvios relativos médios das heurísticas tendem a diminuir com o aumento de n . Em especial, pode-se constatar que o desempenho das heurísticas que adotam o mecanismo de inserção NEH ou similar tendem a obter desempenhos individuais muito próximos nas instâncias de 200 e 500 tarefas. No cenário YS, pode ser observado também que, a medida que n aumenta, MME-A, MME-B, NHPF1 e NHPF2 tendem a gerar soluções piores que as geradas por MMA, MMB, HPF1 e HPF2, respectivamente. No cenário RGM, ao passo que tPF-NEH(x), MM-A, MM-B, HPF1 e HPF2 apresentam melhoria do desempenho a medida que n aumenta, MME-A, MME-B, NHPF1, NHPF2 e tPF-NEH obtiveram desvios relativos maiores. Já para o número de máquinas (m), pode-se constatar na Figura 82 que as heurísticas também tendem a obter desvios relativos menores com o aumento de m .

Figura 82 - Desempenho das heurísticas construtivas em relação à m para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Assim como ocorre em n , na comparação entre heurísticas, é possível observar que métodos que adotam o mesmo mecanismo de inserção tendem a obter desempenhos muito similares para qualquer valor de m , como é o caso de MM-A e MM-B, HPF1 e HPF2 e MME-A, MME-B, tPF-NEH, NHPF1 e NHPF2.

Figura 83 - Desempenho das heurísticas construtivas em relação à duração do *setup* para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Na Figura 83 é evidenciado que há diferenças relevantes de desempenho das heurísticas, em especial, entre heurísticas que adotam diferentes mecanismos de construção da solução. Pode-se notar que as versões de tPF-NEH(x), MM-A, MM-B, HPF1 e HPF2 obtiveram o melhor desempenho quando a duração do *setup* corresponde a até 25% dos tempos de

processamento e tendem a piorar de desempenho com o aumento da duração do *setup*. Já as heurísticas que adotam o mecanismo de inserção NEH apresentam desempenhos individuais muito similares em todas as faixas de *setup* avaliadas. Pelos gráficos a) e b), mostrados na Figura 83, observa-se que as heurísticas que adotam o mecanismo de inserção NEH tendem a obter desvios relativos muito próximos independente do valor do *setup*, o que sugere que MME-A, MME-B, NHPF1, NHPF2 e tPF-NEH tendem a serem estáveis em relação aos tempos de *setup*. Por outro lado, fica evidenciado que os desempenhos de tPF-NEH(x), MM-A, MM-B, HPF1 e HPF2 tendem a piorar com o aumento do tempo de *setup*.

Figura 84 - Desempenho das heurísticas construtivas em relação aos níveis de manutenção preventiva para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Os gráficos a) e b), apresentados na Figura 84, evidenciam que as heurísticas apresentam desvios relativos médios menores a medida que os níveis de MP (l) aumentam. Em geral, as heurísticas tendem a obter desempenhos similares nos dois cenários avaliados. De forma análoga ao que ocorre com os parâmetros anteriores, heurísticas com o mesmo mecanismo de inserção tendem a obter desvios relativos próximos. O mesmo não se pode dizer no desempenho individual das heurísticas em relação aos níveis de MP. Como evidenciado no gráfico b), MME-A, MME-B, tPF-NEH, NHPF1 e NHPF2 apresentam desvios cada vez menores com o aumento de l . Já tPF-NEH(x), MM-A, MM-B, HPF1 e HPF2 tendem a obter desempenhos muito próximos para valores de l a partir de 2.

Tabela 38 - Tempo computacional (em segundos) das heurísticas construtivas no cenário YS

n	HPF1	HPF2	NHPF1	NHPF2	tPF-NEH	tPF-NEH(1)	tPF-NEH(2)	tPF-NEH(5)	MM-A	MM-B	MME-A	MME-B
20	0,001	0,001	0,004	0,005	0,005	0,004	0,008	0,020	0,001	0,001	0,003	0,003
50	0,008	0,007	0,045	0,047	0,045	0,034	0,069	0,173	0,006	0,006	0,038	0,038
100	0,051	0,045	0,387	0,387	0,390	0,195	0,390	0,974	0,034	0,034	0,339	0,348
200	0,382	0,336	3,942	3,968	4,000	1,210	2,414	6,049	0,225	0,225	3,396	3,441
500	5,660	5,280	88,639	88,411	83,094	13,390	26,829	67,083	3,126	3,126	69,689	69,157
Média	1,220	1,134	18,603	18,564	17,507	2,967	5,942	14,860	0,678	0,678	14,693	14,597

Fonte: Autor, 2019.

Tabela 39 - Tempo computacional (em segundos) das heurísticas construtivas no cenário RGM

n	HPF1	HPF2	NHPF1	NHPF2	tPF-NEH	tPF-NEH(1)	tPF-NEH(2)	tPF-NEH(5)	MM-A	MM-B	MME-A	MME-B
20	0,001	0,001	0,004	0,005	0,005	0,004	0,008	0,020	0,001	0,001	0,003	0,003
50	0,008	0,007	0,045	0,047	0,045	0,034	0,069	0,173	0,006	0,006	0,038	0,038
100	0,051	0,045	0,387	0,387	0,390	0,195	0,390	0,974	0,034	0,034	0,339	0,348
200	0,360	0,317	3,719	3,743	3,773	1,141	2,277	5,706	0,212	0,212	3,204	3,246
500	5,340	4,981	83,622	83,407	78,391	12,632	25,311	63,286	2,949	2,949	65,744	65,242
Média	1,152	1,070	17,555	17,518	16,521	2,801	5,611	14,032	0,640	0,640	13,866	13,775

Fonte: Autor, 2019.

Conforme salientado nas Tabelas 38 e 39, as heurísticas mais eficientes para o problema são MM-A, MM-B, HPF1 e HPF2, embora não sejam as mais eficazes. Pode-se notar que, mesmo requerendo mais tempo computacional em comparação com MM-A e MM-B, tPF-NEH(x) obtém desempenho em qualidade de solução superior a todas as outras heurísticas, em tempo computacional médio menor que NHPF1, NHPF2, tPF-NEH, MME-A e MME-B. Nas Figuras 85 e 86 são apresentados os *trade-offs* entre eficiência e eficácia das heurísticas avaliadas nos cenários YS e RGM, respectivamente.

Figura 85 - Desempenho das heurísticas construtivas em relação ao *trade-off* entre tempo computacional e desvio relativo no cenário YS

Fonte: Autor, 2019.

Figura 86 - Desempenho das heurísticas construtivas em relação ao *trade-off* entre tempo computacional e desvio relativo no cenário RGM

Fonte: Autor, 2019.

Pelo que se pode observar, todas as heurísticas apresentam tempo computacional viável. Logo, o tomador de decisão pode optar por escolher a heurística mais adequada de acordo com os recursos disponíveis. Conclui-se também que MM-A, MM-B e as versões de tPF-NEH(x) compõem a curva de pareto. Neste sentido, cabe ao programador, dado os recursos disponíveis, qual heurística será utilizada.

Pelos resultados apresentados anteriormente, a heurística tPF-NEH(x) obteve os melhores desempenhos nos dois cenários YS e RGM, com diferença estatística significativa em comparação as outras heurísticas e em tempo computacional viável. Por estas razões, tPF-NEH(5) será adotada como *input* para as metaheurísticas propostas para solução do problema.

5.3.2 Calibração de parâmetros das metaheurísticas GRASP(HPF2), ABC_RCT e VBIH

Nesta Seção são apresentados os resultados da calibração de parâmetros para as metaheurísticas adaptadas para o problema $F_m | block, NM_j^t, s_{j,k} | (\sum_i C_i | MP_{cost} = C)$. Para análise do *design* de experimentos, utilizou-se a base de instância de problemas para ajuste de parâmetros, descrita na Seção 3.6. Cada versão do algoritmo avaliado foi rodado de forma independente três vezes. O melhor resultado encontrado desses três experimentos foi tido como a solução do problema da dada versão. O tempo estabelecido para todos os métodos foi de $n \times m \times l \times 30$ milissegundos. Conforme já relatado na Seção 5.3.1, tPF-NEH(5) foi adotado como *input* para as metaheurísticas. Os resultados foram avaliados estatisticamente por meio da aplicação da ANOVA, avaliando a hipótese de existir diferença estatística significativa, ao nível de 5% de significância, entre os parâmetros necessários para cada heurística, bem como a interação entre diferentes parâmetros.

5.3.2.1 Parametrização de GRASP(HPF2)

Um estudo computacional foi realizado para identificar e definir os melhores valores para $(min\mu; max\mu)$. Os intervalos de $(min\mu; max\mu)$ propostos por Ribas e Companys (2015) foram utilizados para calibração dos parâmetros. Os autores consideram oito faixas de valores para $(min\mu; max\mu)$: [0, 1], [0,1, 1], [0,1, 1], [0,3, 1], [0,4, 1], [0,5, 1], [0,6, 0,9] e [0,7, 0,8].

Neste sentido, oito versões de GRASP(HPF2) foram testadas. Os resultados da ANOVA para a interação entre os parâmetros estabelecidos para GRASP(HPF2) nos cenários YS podem ser vistos nas Tabelas 40 e 41.

Tabela 40 - ANOVA para análise dos parâmetros adotados em HPF2(GRASP) no cenário YS

Fator	Soma dos Quadrados	Graus de Liberdade	Quadrado Médio	Valor de F	Valor de p
$(\min\mu; \max\mu)$	7,122	7	1,017	0,617	0,742
n	318,036	4	79,509	48,221	0,000
m	124,365	3	41,455	25,142	0,000
$(\min\mu; \max\mu) * n$	31,363	28	1,120	0,679	0,897
$(\min\mu; \max\mu) * m$	7,146	21	0,340	0,206	1,000
$n * m$	319,662	12	26,639	16,156	0,000

Fonte: Autor, 2019.

Tabela 41 - ANOVA para análise dos parâmetros adotados em HPF2(GRASP) no cenário RGM

Fator	Soma dos Quadrados	Graus de Liberdade	Quadrado Médio	Valor de F	Valor de p
n	1853,714	4	463,428	278,720	0,000
$(\min\mu; \max\mu)$	4,372	7	0,625	0,376	,917
m	100,206	3	33,402	20,089	0,000
$m * (\min\mu; \max\mu)$	13,588	28	0,485	0,292	1,000
$n * m$	1065,955	12	88,830	53,425	0,000
$(\min\mu; \max\mu) * n$	15,634	21	0,744	0,448	0,986

Fonte: Autor, 2019.

Os resultados da ANOVA para os dois cenários revelam que não há diferença estatística entre todas as faixas de $(\min\mu; \max\mu)$ consideradas, bem como a interação entre $n * (\min\mu; \max\mu)$ e $(\min\mu; \max\mu) * m$. Neste sentido, foram plotados os gráficos de médias apresentados nas Figuras 87 e 88.

Figura 87 - Desvios relativos da interação entre n e $(\min\mu; \max\mu)$ no cenário YS

Fonte: Autor, 2019.

Figura 88 - Desvios relativos da interação entre n e $(\min\mu; \max\mu)$ no cenário YS

Fonte: Autor, 2019.

Pelos gráficos apresentados nas Figuras 87 e 88 pode-se constatar diferentes comportamentos em relação aos valores de $(\min\mu; \max\mu)$ para cada valor de n . No cenário YS,

embora não haja diferença estatística entre as faixas de μ , pode-se notar que para $n = 40$, a faixa (0,5, 1) gerou a menor média do desvio relativo. Para $n = 80, 120$ e 160 , pode-se notar que GRASP(HPF2) tende a se desempenhar melhor para as faixas mais amplas de μ , ou seja, (0, 1) e (0,1, 1). Já para $n = 200$, é visto que o algoritmo atinge o menor desvio relativo na faixa (0,6, 0,9). Já para o cenário RGM, pode-se observar que o algoritmo obtém o menor desvio relativo para $n = 40$ na faixa (0,1, 1). Para $n = 80, 120$ e 160 é observado que a faixa (0,2, 1) obtém o melhor desempenho. Já para $n = 200$, obtém-se os melhores desempenhos para a faixa (0,3, 1).

Dos resultados apresentados anteriormente, os seguintes valores apresentados na Tabela 42 podem ser estabelecidos para os intervalos de tarefas.

Tabela 42 - Faixas de valores para $(\min\mu; \max\mu)$ nos cenários YS e RGM

n	Cenário YS	Cenário RGM
$n \leq 40$	(0,5, 1)	(0,1, 1)
$40 < n \leq 160$	(0, 1)	(0,2, 1)
$n > 160$	(0,6, 0,9)	(0,3, 1)

Fonte: Autor, 2019.

5.3.2.1 Parametrização de DABC_RCT

O algoritmo DABC_RCT necessita de dois parâmetros a serem calibrados: População e o número de vezes que o procedimento TNO é executado. Por meio de experimentações computacionais, Ribas e Companys (2015) estabelecem três faixas de valores para os dois parâmetros, que também foram adotadas para o *design* de experimentos desta seção. Portanto, foram testadas como número de indivíduos $N = 5, 6$, e 7 e $t = 2, 3$ e 4 . Utilizando-se as bases de instâncias de problemas para ajustes de parâmetros, descritas na Seção 3.6, no total, 9 versões do algoritmo DABC_RCT foram testadas. Pelos resultados apresentados na Seção 5.3.1, o primeiro indivíduo da população é gerado pela heurística tPF-NEH(5) e o restante das soluções é gerado de acordo com os parâmetros de $(\min\mu; \max\mu)$ encontrados durante a calibração dos parâmetros de GRASP(HPF2). Os resultados da aplicação da ANOVA para os cenários YS e RGM podem ser vistos nas Tabelas 43 e 44. Foi considerado um nível de significância de 5% e o desvio relativo como variável dependente.

Tabela 43 - Resultados da ANOVA ao nível de 5% de significância dos parâmetros necessários para DABC_RCT no cenário YS

Fator	Soma dos Quadrados	Graus de Liberdade	Quadrado Médio	Valor de F	Valor de <i>p</i>
População	5,464	2	2,732	0,606	0,546
TNO	7,677	2	3,838	0,851	0,427
População * TNO	8,238	4	2,060	0,457	0,767

Fonte: Autor, 2019.

Tabela 44 - Resultados da ANOVA dos parâmetros necessários para DABC_RCT no cenário YS

Fator	Soma dos Quadrados	Graus de Liberdade	Quadrado Médio	Valor de F	Valor de <i>p</i>
População	16,291	2	8,145	1,754	0,173
TNO	5,384	2	2,692	0,580	0,560
População * TNO	7,645	4	1,872	0,403	0,698

Fonte: Autor, 2019.

Como pode ser observado, em ambos os cenários, não houve diferença estatística em nenhum dos parâmetros avaliados e nem da interação entre População e o valor de *t* (representado como TNO). Um teste de Tukey também foi aplicado aos dados gerados pelo *design* de experimentos, afim de verificar a existência de subconjuntos das médias dos desvios relativos para cada parâmetro avaliado. Como já identificado pela ANOVA, em ambos os cenários, os dois parâmetros apresentaram apenas um subconjunto homogêneo.

Para o cenário YS, o subconjunto População = {5, 6, 7} apresentou médias do desvio relativo iguais a 1,517%, 1,549% e 1,599%, respectivamente, e o subconjunto TNO = {3, 1, 2} apresentou médias iguais a 1,541%, 1,542% e 1,583%. Para o cenário RGM, foi constatado que o subconjunto População = {5, 6, 7} obteve médias correspondentes a 1,649%, 1,667% e 1,697%, respectivamente e o subconjunto TNO {1, 2, 3} apresentaram médias aproximadas de 1,642%, 1,673% e 1,699%, respectivamente. Os resultados encontrados pela análise ANOVA e pelo teste de honestidade de Tukey são reforçados visualmente pelos gráficos apresentados nas Figuras 89 e 90.

Figura 89 - Desvios relativos médios para os valores adotados para a População nos cenários: a) YS e b) RGM

Fonte: Autor, 2019

Pelos gráficos a) e b) apresentados na Figura 89, pode-se notar que nos dois cenários avaliados, a adoção de cinco indivíduos para o parâmetro População obteve o melhor resultado. Já na Figura 90, pode-se notar pequenas diferenças entre os cenários YS (a) e RGM (b). Os dois gráficos apresentados sugerem que, apesar de não apresentarem diferenças estatísticas, o valor de $t = 3$ apresentou o melhor resultado.

Figura 90 - Desvios relativos médios para os valores adotados para a População nos cenários: a) YS e b) RGM

Fonte: Autor, 2019

Pelas análises realizadas anteriormente, adotou-se a População = 5 e TNO = 3 para os dois cenários avaliados.

5.3.2.1 Parametrização de VBIH

Em VBIH, três parâmetros são necessários para o funcionamento da metaheurística: o tamanho máximo do bloco (bS), o número de movimentos feitos pelo bloco ($mSmax$) e o parâmetro de ajuste da temperatura (τP). As configurações utilizadas por Tasgetiren et al. (2016) para cada parâmetro foram adotadas para *design* dos experimentos, ou seja, $bS \in \{4, 8, 12, 16\}$, $mSmax \in \{0,1*(n - bS), 0,2*(n - bS), 0,3*(n - bS)\}$ e $\tau P \in \{0,1, 0,2, 0,3, 0,4, 0,5\}$. No total, $4 \times 3 \times 5 = 60$ configurações de VBIH foram testadas no banco de instâncias para ajuste de parâmetros. Foi estabelecido como critério de parada um tempo de $n \times m \times l \times 30$ milissegundos. A heurística tPF-NEH(5) foi utilizada como *input* para solução inicial do algoritmo. Os resultados da ANOVA para as interações entre os parâmetros adotados podem ser vistos nas Tabelas 45 e 46.

Tabela 45 - Resultados da ANOVA ao nível de 5% de significância dos parâmetros necessários para VBIH no cenário YS

Fator	Soma dos Quadrados	Graus de Liberdade	Quadrado Médio	Valor de F	Valor de p
bS	178,411	3	59,470	8,059	0,000
$mSmax$	375,925	2	187,963	25,471	0,000
τP	33,808	4	8,452	1,145	0,333
$bS * \tau P$	63,663	12	5,305	0,719	0,734
$bS * Smax$	9,360	6	1,560	0,211	0,973
$\tau P * Smax$	14,092	8	1,761	0,239	0,984

Fonte: Autor, 2019.

Tabela 46 - Resultados da ANOVA ao nível de 5% de significância dos parâmetros necessários para VBIH no cenário RGM

Fator	Soma dos Quadrados	Graus de Liberdade	Quadrado Médio	Valor de F	Valor de p
bS	213,852	3	71,284	12,460	0,000
$mSmax$	295,837	2	147,918	25,854	0,000
τP	30,146	4	7,537	1,317	0,261
$Bsmax * Smax$	30,436	6	5,073	0,887	0,504
$Bsmax * \tau P$	31,116	12	2,593	0,453	0,942
$Smax * \tau P$	32,881	8	4,110	0,718	0,676

Fonte: Autor, 2019.

Como pode ser constatado, para os dois cenários avaliados, a ANOVA identificou diferença estatística significativa ao nível de 5% de significância para os parâmetros bS e $mSmax$. Um teste de honestidade de Tukey foi rodado para os resultados encontrados nos cenários YS e RGM para os dois fatores avaliados.

Para o cenário YS, o teste de Tukey identificou dois subconjuntos homogêneos em relação ao parâmetro bS : o primeiro subconjunto obtém $bS = (4, 12, 8)$ com médias iguais a 2,004%, 2,045% e 2,047% respectivamente e o subconjunto contém $bS = 16$ com média de 2,214%. Em relação ao parâmetro $mSmax$, o teste de Tukey identificou três subconjuntos, ou seja, os três valores de $mSmax$ apresentam diferenças estatísticas entre eles, sendo que $mSmax = 0,3*(n - bS)$ obtém média de 1,983%, $mSmax = 0,2*(n - bS)$ corresponde à média 2,046% e $mSmax = 0,1*(n - bS)$ possui média de 2,136%.

Para o cenário RGM, o teste de Tukey identificou três subconjuntos homogêneos dentro do conjunto bS : $bS = (4, 8)$ com médias de 2,283% e 2,308%, respectivamente; $bS = (8, 12)$ com respectivas médias iguais 2,308% e 2,341% e $bS = 16$ com média correspondente a 2,409%. Para o parâmetro $mSmax$ foram observados dois subconjuntos: $mSmax = \{0,3*(n - bS), 0,2*(n - bS)\}$ com médias de 2,293% e 2,299%, respectivamente e $mSmax = 0,1*(n - bS)$ com média de 2,413%.

Na Figura 91 é apresentado o comportamento da heurística em relação ao tamanho máximo do bloco de tarefas, bS , nos dois cenários YS (a) e RGM (b).

Figura 91 - Desvios relativos médios para os valores adotados para bS nos cenários: a) YS e b) RGM

Fonte: Autor, 2019

Apesar das diferenças de desempenho em relação a $bS = 8$ e 12, os gráficos a) e b) apresentados na Figura 91 sugerem que VBIH obtém seu melhor desempenho quando $bS = 4$.

As diferenças de desempenho de VBIH entre os cenários YS e RGM para o parâmetro $mSmax$ podem ser vistos nos gráficos a) e b) mostrados na Figura 92.

Figura 92 - Desvios relativos médios para os valores adotados para $mSmax$ nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Em ambas as políticas pode ser observado que VBIH apresenta seu melhor desempenho na situação em que $mSmax = 0,3 \cdot (n - bS)$. Já em relação ao parâmetro τP , pode ser observado nos gráficos a) e b) da Figura 93 que nos dois cenários avaliados, VBIH apresenta o melhor desempenho médio para $\tau P = 0,1$.

Figura 93 - Desvios relativos médios para os valores adotados para τP nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Por apresentarem desempenhos similares em ambos os cenários avaliados, dos resultados apresentados anteriormente, as seguintes configurações serão utilizadas como parâmetros fixos do VBIH para os dois cenários YS e RGM: $bS = 4$, $mSmax = 0,3 \cdot (n - bS)$ e $\tau P = 0,1$.

5.3.3 Análise dos resultados das metaheurísticas para minimização do tempo total de fluxo para problemas de pequeno porte

Nesta seção são apresentados os resultados do desempenho das metaheurísticas em comparação às soluções dadas pelo IBM CPLEX. Foram comparados também o efeito da busca local proposta para o reposicionamento das atividades de manutenção preventiva (MP) em uma dada sequência. Assim como ocorreu na minimização da duração total da programação na Seção 4, a busca local foi aplicada apenas para a solução final dada pela metaheurística. Neste sentido, a solução final gerada pelas metaheurísticas são rotuladas como DABC_RCT Padrão, GRASP(HPF2) Padrão e VBIH Padrão, e a aplicação da busca local às soluções finais dadas pelas metaheurísticas são rotuladas como DABC_RCT BL, GRASP(HPF2) BL e VBIH BL, respectivamente. Todas as configurações encontradas como as melhores nas subseções que trataram da parametrização das metaheurísticas são utilizadas nesta seção. Foi estabelecido como critério de parada o tempo computacional de $n \times m \times l \times 30$ milisegundos. A heurística tPF-NEH(5) foi utilizada para gerar a primeira fonte de alimento de DABC_RCT e a primeira solução de VBIH, além de estabelecer o valor da restrição *total_cost* para todas as metaheurísticas.

O CPLEX encontrou soluções para até $n = 8$, $m = 5$ e $l = 2$ no cenário YS, dentro de um limite de 3.600 segundos por problema. Para instâncias em que $n = 6$ e 7 e $l > 2$, o MILP encontrou solução para a maior parte dos 40 problemas por instância, com exceção de $n = 7$, $m = 5$ e $l = 4$, onde foi obtida solução de 6 problemas. Dos 2560 problemas de pequeno porte, o CPLEX conseguiu solucionar 2394 problemas no cenário YS.

Para o cenário RGM, o CPLEX conseguiu encontrar solucionar 2480 problemas dentro do tempo limite de 3.600 segundos para cada problema. A partir de $n = 6$, $m > 3$ e $l > 2$, o CPLEX encontrou a solução da maior parte das instâncias, exceto para $n = 7$, $m = 5$ e $l = 4$, onde foram encontradas 15 soluções dos 40 problemas da instância. Os resultados do desempenho das heurísticas pela combinação $n \times m$ nos cenários YS e RGM podem ser vistos nas Tabelas 47 e 48, respectivamente.

Tabela 47 - Desempenho das metaheurísticas para problemas de pequeno porte no cenário YS, com desvios relativos médios agrupados por $n \times m$

n	m	Modelo	GRASP(HPF2)	GRASP(HPF2)	DABC_RCT	DABC_RCT	VBIH	VBIH Busca
			Padrão	Busca Local	Padrão	Busca Local	Padrão	Local
5	2	0	1,603	0,602	1,575	0,322	1,629	0,376
	3	0	9,736	8,538	11,192	9,613	11,096	9,515
	4	0	13,332	12,263	16,477	14,398	16,331	14,257
	5	0	14,926	13,339	17,952	15,923	17,712	15,753
6	2	0	2,347	1,333	1,800	0,754	1,470	0,528
	3	0	13,267	11,775	14,438	12,687	14,117	12,447
	4	0	17,085	15,664	20,055	17,967	19,453	17,207
	5	0	18,169	16,577	21,013	18,866	20,521	18,481
7	2	0	3,548	2,697	3,056	2,132	2,645	1,603
	3	0	14,396	12,898	15,656	13,825	15,168	13,356
	4	0	20,369	18,351	22,172	20,199	21,748	19,679
	5	0	23,506	21,695	25,248	23,580	24,742	22,956
8	2	0	4,154	3,009	2,789	1,538	2,532	1,227
	3	0	15,764	14,084	16,308	14,621	15,368	13,827
	4	0	21,815	20,059	21,756	20,141	21,363	19,712
	5	0	24,635	23,497	25,036	23,851	25,163	24,086
Média		0	13,666	12,274	14,783	13,151	14,441	12,813
Geral								

Fonte: Autor, 2019.

Pela média geral, constata-se que GRASP(HPF2) obteve a melhor média do desvio relativo nos dois cenários. Em especial para $m > 2$, a referida metaheurística apresenta as menores médias do desvio relativo. VBIH tende a apresentar melhorias quando $n = 8$, em ambos os cenários. Pode-se constatar também pelos resultados apresentados nas Tabelas 47 e 48, que a aplicação da busca local proposta reduz significativamente o desvio relativos médios em relação a solução dada pelas versões Padrões. Em geral, a busca local diminuiu em cerca de 1,392% e 1,084% o desvio relativo da metaheurística GRASP(HPF2) nos cenários YS e RGM, respectivamente. Para DABC_RCT, a melhoria apresentada pela busca local resultou em 1,632% no cenário YS e 1,166% no cenário RGM e em VBIH, obteve-se uma diminuição de 1,628% e 1,151% para YS e RGM, respectivamente.

Tabela 48 - Desempenho das metaheurísticas para problemas de pequeno porte no cenário RGM, com desvios relativos médios agrupados por $n \times m$

n	m	Modelo	GRASP(HPF2)	GRASP(HPF2)	DABC_RCT	DABC_RCT	VBIH	VBIH Busca
			Padrão	Busca Local	Padrão	Busca Local	Padrão	Local
5	2	0	0,612	0,230	0,479	0,044	0,479	0,044
	3	0	9,642	8,897	10,551	9,788	10,366	9,580
	4	0	13,600	12,639	15,244	14,133	15,003	13,912
	5	0	15,730	14,470	17,975	16,527	17,816	16,321
6	2	0	1,127	0,671	0,924	0,450	0,756	0,248
	3	0	11,333	10,480	12,304	11,233	12,090	11,037
	4	0	14,450	13,300	16,408	14,958	16,562	15,238
	5	0	18,704	17,116	21,364	19,527	21,007	19,235
7	2	0	2,002	1,273	1,576	0,688	1,291	0,454
	3	0	12,738	11,490	13,549	12,409	13,061	12,002
	4	0	18,285	16,775	19,285	17,699	19,191	17,507
	5	0	22,194	20,746	24,493	23,043	24,194	22,720
8	2	0	2,370	1,784	1,705	1,103	1,339	0,730
	3	0	14,166	12,949	14,883	13,379	14,649	13,199
	4	0	21,955	20,204	21,706	20,305	21,387	19,937
	5	0	24,909	23,457	24,927	23,430	24,360	22,966
Média Geral		0	12,739	11,655	13,586	12,420	13,347	12,196

Fonte: Autor, 2019.

O teste de honestidade de Tukey ao nível de 5% de significância foi aplicado aos desvios relativos das heurísticas para os cenários YS e RGM. Os resultados dos testes de Tukey estão dispostos nas Tabelas 49 e 50.

Tabela 49 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário YS

Métodos	N	Subconjuntos para $\alpha = 0,05$				
		1	2	3	4	5
Modelo	2.391	0,000				
GRASP Busca Local	2.391		11,616			
VBIH Busca Local	2.391		12,144	12,144		
DABC_RCT Busca Local	2.391		12,493	12,493		
GRASP Padrão	2.391			13,001	13,001	
VBIH Padrão	2.391				13,784	13,784
DABC_RCT Padrão	2.391					14,135
Significância		1,000	0,166	0,188	0,287	0,955

Fonte: Autor, 2019.

Tabela 50 - Teste de honestidade de Tukey ao nível de 5% de significância das metaheurísticas para problemas de pequeno porte no cenário RGM

Métodos	N	Subconjuntos para $\alpha = 0,05$				
		1	2	3	4	5
Modelo	2.477	0,000				
GRASP Busca Local	2.477		11,324			
VBIH Busca Local	2.477		11,864	11,864		
DABC_RCT Busca Local	2.477		12,083	12,083	12,083	
GRASP Padrão	2.477			12,394	12,394	12,394
VBIH Padrão	2.477				13,005	13,005
DABC_RCT Padrão	2.477					13,238
Significância		1,000	0,240	0,677	0,075	0,136

Fonte: Autor, 2019.

Em ambos os cenários, o teste de Tukey identificou cinco subconjuntos homogêneos, ou seja, grupos de metaheurísticas cujas médias do desvio relativo não apresentam diferença estatística significativa ao nível de 5% de significância. Considerando a comparação entre cenários, uma pequena diferença pode ser observada na composição do subconjunto 5. No geral, GRASP(HPF2) obteve os melhores resultados, seguidos de VBIH e DABC_RCT, em todos os cenários avaliados e também para as versões Padrão e as respostas dadas pela Busca Local. Estes resultados também podem ser visualizados nos gráficos de médias a) e b) mostrados na Figura 94. Pode-se constatar que no cenário YS, a aplicação da busca local às soluções geradas pelas metaheurísticas, apresentam diferença estatística significativa em comparação com suas versões Padrões, ou seja, as diferenças médias de GRASP BL, VBIH BL e DABC_RCT BL são significativas em comparação com GRASP Padrão, VBIH BL e DABC_RCT BL, respectivamente. Já no cenário RGM, as médias entre as versões BL e Padrão tendem a serem mais próximas, em especial DABC_RCT BL e DABC_RCT Padrão, como constatado tanto pela Figura 94.b) quanto pela Tabela 50.

Figura 94 - Gráfico de médias das metaheurísticas ao nível de 5% de significância para problemas de pequeno porte nos cenários: a)YS e b) RGM

Fonte: Autor, 2019.

Em relação ao número de tarefas, embora hajam diferenças no desempenho das heurísticas para $n = 8$, pode-se constatar que todos métodos avaliados tendem a obter desempenhos similares em ambos os cenários. Por exemplo, tanto GRASP Padrão como GRASP BL apresentam o mesmo comportamento crescente a medida que n aumenta. Neste sentido, pode-se constatar diferenças de desempenho mais expressivas entre VBIH e DABC_RCT com o aumento de n .

Figura 95 – Desempenho das metaheurísticas em relação ao número de tarefas para problemas de pequeno porte nos cenários: a)YS e b) RGM

Fonte: Autor, 2019.

Já em relação ao número de máquinas (m), é observado que todos os métodos avaliados tendem a piorar de desempenho, significativamente, com o aumento de m . Pode-se observar

que todos os métodos tendem a obter médias muito próximas umas das outras, principalmente para $m = 5$ e $m = 10$, onde se observa que há pouca diferença entre as versões Padrão e a Busca Local. Nos dois cenários avaliados YS e RGM, é observado similaridade no comportamento das heurísticas.

Figura 96 – Desempenho das metaheurísticas em relação ao número de máquinas para problemas de pequeno porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Conforme pode ser visto pelos gráficos a) e b) mostrados na Figura 97, é observado que no cenário YS, todas as metaheurísticas tendem a se desempenhar melhor a medida que a duração do *setup* aumenta. No geral todas as metaheurísticas tem sua menor média do desvio relativo quando a duração do *setup* corresponde a até 125% dos tempos de processamento. Uma diferença em relação à supracitada duração do *setup* pode ser vista no cenário RGM. Neste sentido, pode-se observar que para cenários de MP onde são considerados intervalos menos espaçados, as metaheurísticas tendem a obter melhores desempenhos com duração de *setups* mais longos.

Figura 97 – Desempenho das metaheurísticas em relação à duração do *Setup* para problemas de pequeno porte nos cenários: a)YS e b) RGM

Fonte: Autor, 2019.

Já em relação aos níveis de manutenção (l), conforme pode ser visto nos gráficos a) e b) da Figura 98, observa-se que a aplicação da busca local tende a gerar melhorias significativas nas soluções Padrões dadas pelas metaheurísticas. Pode-se notar que a busca local tende a contribuir de maneira mais significativa com o aumento de l . A aplicação da busca local apresenta o melhor desempenho para $l = 4$, em ambos os cenários.

Figura 98 – Desempenho das metaheurísticas em relação aos níveis de manutenção preventiva para problemas de pequeno porte nos cenários: a)YS e b) RGM

Fonte: Autor, 2019.

Os resultados mostrados anteriormente sugerem que a aplicação da busca local tende a ser mais significativa para o número de tarefas, níveis de manutenção preventiva e duração do *setup*. Entretanto, pode-se concluir que, na média geral, a adoção da busca local apresenta melhorias expressivas para as soluções padrões dadas pelas metaheurísticas. Logo, pode-se

considerar a busca local proposta como parte da metodologia de solução do problema $F_m|block, NM_j^t, s_{j,k}|(\sum_i C_i |MP_{cost} = C)$. Em relação as metaheurísticas, pode-se constatar que GRASP(HPF2) obteve o melhor desempenho na média geral. Entretanto, é necessário também que se analise portes de problemas maiores, uma vez que os desvios relativos apresentados pela referida metaheurística tende a aumentar e se distanciar de VBIH para $n = 8$.

5.3.4 Análise dos resultados das metaheurísticas para minimização do tempo total de fluxo para problemas de médio e grande portes

Nesta seção são apresentados os resultados do desempenho das metaheurísticas em relação ao banco de instâncias de problemas de médio e grande porte. A análise dos resultados foi análoga à realizada para os problemas de pequeno porte, ou seja, além da avaliação das soluções finais dadas pelas heurísticas, foram comparados também o impacto da aplicação da busca local proposta para o reposicionamento das atividades de manutenção preventiva (MP) em uma dada sequência. Deste modo, a solução final gerada pelas metaheurísticas são rotuladas como GRASP(HPF2)/GRASP Padrão, DABC_RCT Padrão e VBIH Padrão, e a aplicação da busca local às soluções finais dadas pelas metaheurísticas são rotuladas como GRASP(HPF2)/GRASP Busca Local, DABC_RCT Busca Local e VBIH Busca Local, respectivamente. Foi estabelecido como critério de parada o tempo computacional de $n \times m \times l \times 30$ milisegundos para cada problema. A restrição de custo, as soluções iniciais de GRASP(HPF2) e VBIH e o primeiro indivíduo de DABC_RCT foram gerados pela heurística tPF-NEH(5). Os resultados dos desvios relativos médios agrupados por $n \times m$ para os cenários YS e RGM podem ser vistos nas Tabela 51 e Tabela 52.

Tabela 51 - Desvio relativo médio agrupados por $n \times m$ das metaheurísticas no cenário YS (Continua...)

n	m	GRASP Padrão	GRASP BL	DABC_RCT Padrão	DABC_RCT BL	VBIH Padrão	VBIH BL
20	5	11,810	10,597	3,289	2,601	1,144	0,656
	10	8,371	6,898	3,333	2,320	1,522	0,698
	15	6,799	5,351	3,017	2,059	1,445	0,629
	20	5,612	4,247	3,050	2,100	1,576	0,651
50	5	8,505	8,255	3,546	3,340	0,983	0,837
	10	5,803	5,281	3,991	3,724	1,272	1,061

Tabela 51 - Desvio relativo médio agrupados por $n \times m$ das metaheurísticas no cenário YS (Conclusão.)

n	m	GRASP Padrão	GRASP BL	DABC_RCT Padrão	DABC_RCT BL	VBIH Padrão	VBIH BL
50	15	5,142	4,615	3,798	3,449	1,179	0,895
	20	4,355	3,797	3,777	3,337	1,466	1,126
100	5	2,521	2,452	3,416	3,359	1,787	1,735
	10	1,034	0,899	2,734	2,656	2,287	2,199
	15	0,891	0,745	2,898	2,814	2,237	2,151
	20	0,524	0,366	2,589	2,469	2,255	2,152
200	5	0,509	0,495	2,126	2,114	1,995	1,985
	10	0,098	0,073	1,746	1,723	1,869	1,848
	15	0,048	0,022	1,739	1,713	1,758	1,729
	20	0,061	0,028	1,601	1,571	1,749	1,716
500	5	0,187	0,185	0,729	0,727	0,824	0,822
	10	0,145	0,142	0,829	0,825	0,993	0,990
	15	0,016	0,013	0,752	0,748	0,842	0,838
	20	0,045	0,040	0,885	0,881	0,986	0,981
Média Geral		3,124	2,725	2,492	2,226	1,508	1,285

Fonte: Autor, 2019.

Tabela 52 - Desvio relativo médio agrupados por $n \times m$ das metaheurísticas no cenário RGM

n	m	GRASP Padrão	GRASP BL	DABC_RCT Padrão	DABC_RCT BL	VBIH Padrão	VBIH BL
20	5	11,590	9,406	3,100	2,032	1,646	0,565
	10	9,249	6,530	2,926	1,437	2,200	0,793
	15	8,093	5,741	2,860	1,393	1,948	0,607
	20	6,787	4,429	2,471	1,078	1,938	0,564
50	5	9,740	8,469	2,896	2,200	1,327	0,740
	10	7,877	6,056	3,453	2,387	1,611	0,659
	15	6,863	5,068	3,305	2,149	1,611	0,627
	20	5,888	4,028	3,198	1,967	1,799	0,628
100	5	4,215	3,516	2,389	1,909	1,092	0,635
	10	3,104	1,863	2,979	1,954	1,956	1,127
	15	2,848	1,362	3,077	1,877	2,135	1,083
	20	2,554	1,018	3,250	1,901	2,424	1,240
200	5	1,227	0,787	0,990	0,575	0,865	0,471
	10	1,351	0,507	1,424	0,665	1,537	0,815
	15	1,307	0,298	1,932	1,041	1,890	1,034
	20	1,186	0,108	2,043	1,117	2,047	1,123
500	5	0,318	0,125	0,412	0,242	0,347	0,164
	10	0,517	0,062	0,754	0,430	0,742	0,348
	15	0,598	0,058	0,890	0,432	0,933	0,436
	20	0,667	0,022	1,220	0,871	1,287	0,677
Média Geral		4,299	2,973	2,278	1,383	1,567	0,717

Fonte: Autor, 2019.

Como pode ser constatado, em ambos os cenários, VBIH obteve a menor média geral do desvio relativo. Entretanto, pode-se observar que a partir de 100 tarefas, GRASP(HPF2) tende a obter os menores desvios relativos médios por instância $n \times m$. Devido ao fraco desempenho para instancias de até 50 tarefas, GRASP(HPF2) Padrão obteve o maior desvio relativo médio. No cenário YS é percebido que a aplicação da busca local para todas as metaheurísticas não obteve melhorias tão expressivas quanto no cenário RGM, em especial, a partir de 100 tarefas. Na comparação entre cenários, pode-se perceber nos gráficos a) e b) da Figura 99 algumas diferenças expressivas, na ordem de desempenho das heurísticas.

Figura 99 - Gráfico de médias ao nível de 5% de significância das metaheurísticas em problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

É observado que os resultados gerados pela busca local obtiveram resultados distintos nos dois cenários avaliados. No cenário YS é constatado que, apesar dos resultados médios da busca local obterem diferença estatística significativa em relação os resultados Padrão dos métodos, os ganhos obtidos por ela não foram tão expressivos quanto no cenário RGM. Por exemplo, pelo gráfico b) da Figura 99 pode-se visualizar que DABC_RCT BL obteve uma média menor que VBIH Padrão em RGM, se aproximando de VBIH BL, diferentemente do que ocorre no gráfico de médias a) que representa o cenário YS. Um teste de honestidade de Tukey foi rodado para os dois cenários, no intuito de identificar possíveis subconjuntos homogêneos entre os métodos avaliados. Os resultados do teste de Tukey para os cenários YS e RGM podem ser vistos nas Tabelas 53 e 54.

Tabela 53 - Teste de honestidade de Tukey ao nível de 5% de significância para metaheurísticas em problemas de médio e grande porte no cenário YS

Métodos	N	Subconjunto para $\alpha = 0,05$					
		1	2	3	4	5	6
VBIH BL	3.200	1,284					
VBIH Padrão	3.200		1,508				
DABC_RCT BL	3.200			2,226			
DABC_RCT Padrão	3.200				2,492		
GRASP(HPF2) BL	3.200					2,725	
GRASP(HPF2) Padrão	3.200						3,124
Significância		1,000	1,000	1,000	1,000	1,000	1,000

Fonte: Autor, 2019.

Tabela 54 - Teste de honestidade de Tukey ao nível de 5% de significância para metaheurísticas em problemas de médio e grande porte no cenário RGM

Métodos	N	Subconjunto para $\alpha = 0,05$				
		1	2	3	4	5
VBIH BL	3.200	0,717				
DABC_RCT BL	3.200		1,383			
VBIH Padrão	3.200		1,567			
DABC_RCT Padrão	3.200			2,278		
GRASP(HPF2) BL	3.200				2,973	
GRASP(HPF2) Padrão	3.200					4,299
Significância		1,000	0,117	1,000	1,000	1,000

Fonte: Autor, 2019.

É possível constatar resultados similares aos da Figura 99 nos testes de Tukey apresentados nas Tabelas 53 e 54. No cenário YS, o teste identificou que todas as heurísticas se diferem estatisticamente entre elas, ou seja, foram identificados seis subconjuntos homogêneos de médias. Já no cenário RGM, DABC_RCT BL e VBIH Padrão foram agrupadas em um mesmo subconjunto, resultando em cinco subconjuntos homogêneos de médias.

Figura 100 - Desvio relativo médio das metaheurísticas em relação à n para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Pelos gráficos a) e b) da Figura 100 são mostrados os desempenhos dos métodos em relação ao número de tarefas (n) nos dois cenários avaliados. Na comparação entre cenários pode-se notar que VBIH e DABC_RCT obtiveram desempenhos muito similares a partir de $n = 200$, diferentemente do que ocorre com GRASP(HPF2)Padrão e BL, que obteve as menores médias a partir de $n = 100$ no cenário YS. Em ambos os cenários, a aplicação da Busca Local (BL) tende a obter melhores resultados para problemas de médio porte, como $n = 20$ e $n = 50$. A medida que n aumenta, a diferença entre a versão Padrão e a Busca Local tende a diminuir consideravelmente, principalmente no cenário YS, que admite intervalos entre manutenções preventivas menos espaçadas.

Figura 101 - Desvio relativo médio das metaheurísticas em relação à m para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Em relação ao número de máquinas (m), pode-se perceber que em ambos os cenários a aplicação da busca local à solução dada por VBIH obteve as menores médias do desvio relativo em todos valores de m . Na comparação entre cenários, a Figura 101 evidencia que o desempenho das metaheurísticas tende a se aproximar a medida que m aumenta. Em geral, DABC_RCT e VBIH apresentam desempenhos estáveis com o aumento de m , diferentemente do que ocorre com GRASP(HPF2).

Figura 102 - Desvio relativo médio das metaheurísticas em relação ao *setup* para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

Já em relação à duração do *setup*, é observado que as diferenças do desvio relativo entre as versões Padrão e a Busca Local aplicada tendem a se tornarem mais expressivas no cenário RGM. Em geral, o desempenho da busca local acompanha o comportamento do desempenho das metaheurísticas com o aumento do *tempo* de *setup*. Pode-se notar também pelos gráficos a) e b) da Figura 102 que as metaheurísticas adotam comportamentos similares a medida que a duração do *setup* aumenta, nos dois cenários YS e RGM, ou seja, GRASP(HPF2) tende a obter desvios relativos maiores, DABC_RCT é estável e VBIH apresenta desvios relativos menores.

Na Figura 103 é evidenciado o desempenho dos métodos em relação aos níveis de MP (l) nos cenários YS (Gráfico a) e RGM (Gráfico b). Em geral, é observado que nos dois cenários avaliados, a aplicação da busca local tende a melhorar significativamente a solução obtida pela versão Padrão de todas as metaheurísticas. Essas diferenças acentuam-se mais com o aumento de l . Nos dois cenários, VBIH BL obtém os melhores resultados. Na comparação entre cenários, é observado que o efeito da busca local é maior para o cenário RGM em todas as metaheurísticas avaliadas.

Figura 103 - Desvio relativo médio das metaheurísticas em relação ao nível de manutenção para problemas de médio e grande porte nos cenários: a) YS e b) RGM

Fonte: Autor, 2019.

De acordo com as análises realizadas anteriormente, pode-se constatar que VBIH obteve os melhores resultados gerais, além de apresentar estabilidade em todos os fatores avaliados. Neste sentido, VBIH é sugerida como metaheurística para problemas de médio e grande porte.

5.4 CONSIDERAÇÕES FINAIS DA SEÇÃO 5

Neste capítulo foram propostas e testadas heurísticas construtivas e metaheurísticas para o problema $F_m|block, NM_j^t, s_{j,k}|(\sum_i C_i |MP_{cost} = C)$. Métodos da literatura do problema $F_m|block, |\sum_i C_i$ foram adaptados para o problema estudado. Neste sentido, as heurísticas construtivas mais eficientes da literatura foram testadas no intuito de avaliar o método que gera o menor custo total de manutenção preventiva (MP) para ser utilizada como dado de entrada para as metaheurísticas adaptadas. No total, 10 heurísticas construtivas foram testadas, sendo elas HPF1, HPF2, NHPF1, NHPF2, tPF-NEH(x), tPF-NEH, MM-A, MM-B, MME-A e MME-B. Pode-se constatar que tPF-NEH(5) obteve o melhor desempenho, em tempo computacional menor que outras heurísticas.

Das metaheurísticas da literatura identificadas no Capítulo 2, foram elencadas GRASP(HPF2), DABC_RCT e VBIH, provadas como as mais eficazes para o problema $F_m|block, |\sum_i C_i$. Para problemas de pequeno porte, GRASP(HPF2) obteve o melhor desempenho geral nos dois cenários YS e RGM. Para problemas de médio e grande portes,

VBIH obteve o melhor desempenho, embora GRASP(HPF2) tenha tido os melhores resultados para instancias de grande porte ($n = 200$ e 500) nos dois cenários avaliados.

Como sugestão para estudos futuros, pretende-se também avaliar as metaheurísticas em faixas de tempo maiores que o critério de parada adotado nesta seção. Convém também salientar que a estrutura de GRASP(HPF2) é mais simples que em relação as outras metaheurísticas avaliadas, o que pode trazer maiores ganhos para problemas de grande porte. Neste sentido, a utilização de outras faixas de tempo como critério de parada pode auxiliar numa avaliação mais ampla das metaheurísticas implementadas.

6 CONCLUSÕES

Esta tese teve como foco de pesquisa o problema *flow shop* sem estoque intermediário entre as m máquinas, tempos de *setup* dependente da sequência das tarefas, ocorrência de manutenção preventiva (MP) para minimização da duração total da programação e do tempo total de fluxo, sujeito que os custos de MP não podem ser maior que um dado valor. Os problemas podem ser denotados como $F_m|block, NM_j^t, s_{j,k}|(C_{max}|total_cost = C)$ e $F_m|block, NM_j^t, s_{j,k}|(\sum_i C_i|total_cost = C)$. Dado que manutenção preventiva e *scheduling* estão diretamente relacionados na realidade prática, este trabalho propõe uma forma de integrar as decisões de sequenciamento de tarefas e programação das MP's. Para atingir esta meta, os seguintes objetivos foram estabelecidos na Seção 1 - Introdução:

OBJETIVO 1 - *Revisão e classificação das publicações existentes de scheduling para o problema flow shop com restrição de estoque intermediário.*

Uma revisão de literatura sistemática foi realizada na Seção 2 para o problema *flow shop* com bloqueio, outras restrições adicionais e diversas funções objetivo. A revisão compreendeu o período entre 1969 à Maio de 2019 e resultou em um total de 139 publicações. A maioria das publicações encontradas trataram da proposição de métodos exatos, heurísticas construtivas e metaheurísticas para diversos problemas. Uma sumarização da revisão da literatura pode ser vista nas Tabelas 2, 3, 4, 5, 6 e 7. Dos resultados encontrados foram levantando os métodos heurísticos (construtivos e metaheurísticas) da literatura para os problemas $F_m|block|C_{max}$, $F_m|block, s_{j,k}|C_{max}$ e $F_m|block|\sum_i C_i$, que são os que mais se assemelham ao foco desta tese.

OBJETIVO 2 - *Definição e modelagem matemática do problema referido, bem como formulações e formas de cálculo das funções-objetivo adotados.*

Na Seção 3, o problema foi definido com detalhes em relação ao sequenciamento de tarefas e também à descrição de como a manutenção preventiva foi abordada. Um modelo de programação linear inteira mista (MILP) foi proposto para o problema. Por ainda não ter sido estudado na literatura, formas de cálculo das datas de início e término das tarefas nas máquinas foram desenvolvidos para cálculo da duração total da programação e do tempo total de fluxo. A forma de cálculo também pode ser aplicada a problemas de scheduling com função-objetivo de atraso total.

OBJETIVO 3 - *Desenvolvimento de procedimentos para programação e reposicionamento das atividades de MP's na sequência de tarefas.*

Na Seção 3.4 foi desenvolvido um procedimento para calcular o desgaste da máquina de acordo com o nível de manutenção e a máquina no intuito de identificar em que posição da sequência de tarefas a manutenção preventiva deve ser realizada. Neste sentido, o procedimento proposto é usado conjuntamente com o cálculo da função-objetivo, uma vez que é necessário descobrir as posições em que as MP's estão para determinar a data de início e término das tarefas. A forma de cálculo desenvolvida também fornece o custo total de MP (*total_cost*) da sequência. Uma busca local foi desenvolvida para reposicionamento das atividades de MP que leva em conta as restrições do *total_cost* e também a possibilidade de reposicionamento dada a condição da máquina no nível avaliado. Esta busca local tem como objetivo minimizar a função-objetivo adotada, avaliando se as atividades de MP's podem ser designadas entre outras tarefas da sequência. Dois cenários de atividades de manutenção preventiva, denominados YS e RGM, foram adotados para gerar o intervalo entre as manutenções preventivas.

OBJETIVO 4 - *Proposição de técnicas heurísticas para o problema abordado com a minimização da duração total da programação do problema e do tempo total de fluxo, dado que o custo de MP não pode ser maior que um determinado valor.*

Visto que a realização da revisão da literatura possibilitou a investigação do estado da arte na área, os métodos heurísticos construtivos encontrados foram adaptados para minimização do custo total de MP para os dois problemas estudados e as melhores metaheurísticas para cada um dos problemas foram escolhidas para solução da função hierárquica. A busca local desenvolvida para reposicionamento das manutenções preventivas, considerada uma heurística, também foi avaliada experimentalmente.

OBEJTIVO 4.1 – *Proposição e avaliação de métodos de solução para o problema*
 $F_m | block, NM_j^t, s_{j,k} | (C_{max} | total_cost = C)$

A Seção 4 apresentou métodos heurísticos construtivos e metaheurísticas para o problema citado. Ao todo, foram adaptadas 14 heurísticas construtivas da literatura para o problema de minimização da duração total da programação (C_{max}) dado custo total de MP (*total_cost*): PF, PFE, MM, MM-B, MME, MME-B, PF-NEH(x), wPF-NEH(x), PW-NEH(x), PFT-NEH(x), PFT-RZ(x), PF-NEH(λ), WPT-NEH e mNEH. Versões alterantivas das oito últimas heurísticas foram também propostas ao problema, de modo que os tempos de *setup*

fossem considerados na decisão do sequenciamento. Todos os métodos levantados foram adaptados ao problema com o objetivo de minimizar o *total_cost* prioritariamente. A minimização de C_{max} foi utilizada como fator de desempate para sequências que gerassem *total_cost* iguais. As 22 heurísticas foram implementadas computacionalmente e avaliadas por meio de testes de estatísticos e análises gráficas. Pode-se constatar que PFT-RZ(5), PFT-RZ(5) e PFT-NEH(5)st obtiveram os melhores desempenho no *trade-off* entre qualidade de solução e tempo computacional. Das metaheurísticas levantadas pela revisão de literatura, duas versões de *Iterated Greedy*, denominadas IG_IJ e IG_RIS e a metaheurística *Discrete Water Wave Optimization* (DWWO) foram adaptadas para o problema proposto com o objetivo de minimizar a duração total da programação dado o *total_cost*. Um *design* de experimentos foi realizado para definir os melhores parâmetros para todas as metaheurísticas adaptadas. Os métodos foram então experimentalmente testados em problemas de pequeno, médio e grande porte. Para problemas de pequeno porte, as metaheurísticas foram comparadas com as soluções geradas pelo MILP proposto e para médio e grande portes, as metaheurísticas foram comparadas entre si. Pode-se constatar que IG_IJ e IG_RIS obtiveram os melhores desempenhos em todas as bases de problemas avaliadas e também nos dois cenários avaliados.

OBJETIVO 4.2 – *Proposição e avaliação de métodos de solução para o problema*
 $F_m | block, NM_j^t, s_{j,k} | (\sum_i C_i | total_cost = C)$

A Seção 5 apresentou métodos de solução para o problema referido. No total, 10 heurísticas construtivas da literatura foram adaptadas para o problema de minimização do tempo total de fluxo: HFP1, HPF2, NHP1, NHPF2, tPF-NEH, tPF-NEH(x), MM-A, MM-B, MME-A e MME-B. As heurísticas foram adaptadas para a função-objetivo de minimização do *total_cost* de modo análogo ao que feito para o problema de minimização de C_{max} . Análises estatísticas comprovaram que tPF-NEH(5) obteve o melhor desempenho nos dois cenários YS e RGM avaliados. As metaheurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP), *Discrete Artificial Bee Colony* (DABC_RCT) e *Variable Block Insertion Heuristic* (VBIH) foram adaptadas para o problema de minimização do tempo total de fluxo dado um valor de *total_cost*. Um *design* de experimentos foi realizado para definir os parâmetros que melhor se ajustam para solucionar o problema proposto. Resultados experimentais para problemas de pequeno porte mostraram que em comparação com o MILP, GRASP(HPF2) obteve os melhores resultados. Para problemas de médio e grande porte os testes estatísticos indicaram que VBIH obteve o melhor desempenho.

OBJETIVO 4.3 – *Eficácia da busca local para reposicionamento das MP's para uma dada sequência aplicada aos problemas $F_m|block, NM_j^t, s_{j,k}|(C_{max}|total_cost = C)$ e $F_m|block, NM_j^t, s_{j,k}|(\sum_i C_i|total_cost = C)$*

A busca local desenvolvida na Seção 3.4 foi aplicada às soluções geradas por todas as metaheurísticas propostas para o problema. Resultados experimentais, apresentados nas Seções 4.3 e 5.3, mostraram que a busca local proposta obteve melhoria significativa em relação às soluções dadas pelas metaheurísticas em todas os portes de problemas avaliados e todos os cenários considerados.

6.1 SUGESTÃO PARA PESQUISAS FUTURAS

Como pesquisas futuras, sugere-se o estudo do problema abordado com outras funções-objetivo como atraso total ou custos atribuídos a atrasos e adiatamentos dado o custo total de MP. Por se tratar de um algoritmo voltado para o reposicionamento das MP's, a busca local proposta também pode ser aplicada para outros objetivos de desempenho.

Como identificado no Capítulo 2, outras oportunidades de pesquisa se concentram em avaliar as restrições adicionais adotadas com outros tipos de bloqueio. Pode-se elencar também o estudo experimental de outras metaheurísticas levantadas pela revisão de literatura, assim como a adaptação de métodos com outras funções-objetivo para o problema proposto.

Por fim, com algumas adaptações, o problema definido nesta tese também pode ser estendido para outros ambientes de produção tais como máquina única, máquinas paralelas, *flow shop* híbrido e *job shop*.

REFERÊNCIAS

ABADI, I. N. K. A new algorithm for minimizing makespan, $C(\max)$, in blocking flow-shop problem through slowing down the operations. **Journal of the Operational Research Society**, v. 58, n. 1, p. 134–140, 2007.

ABADI, I. N. K. I. N. K.; HALL, N. G. .; SRISKANDARAJAH, C. Minimizing Cycle Time in a Blocking Flowshop. **Operations Research**, v. 48, n. 1, p. 177–180, 2000. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/B012369398500534X>>. .

AGGOUNE, R. Minimizing the makespan for the flow shop scheduling problem with availability constraints. **European Journal of Operational Research**, v. 153, n. 3 SPEC. ISS., p. 534–543, 2004.

AGNETIS, A.; PACCIARELLI, D.; ROSSI, F. Batch scheduling in a two-machine flow shop with limited buffer. **Discrete Applied Mathematics**, v. 72, n. 3, p. 243–260, 1997. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-0000698815&partnerID=tZOtx3y1>>. .

AKAY, B.; KARABOGA, D. Artificial bee colony algorithm for large-scale problems and engineering design optimization. **Journal of Intelligent Manufacturing**, v. 23, p. 1001–1014, 2012.

AL-SALEM, M.; KHARBECHÉ, M. Throughput optimization for the Robotic Cell Problem with Controllable Processing Times. **RAIRO - Operations Research**, v. 51, n. 3, p. 805–818, 2016.

ALLAHVERDI, A. The third comprehensive survey on scheduling problems with setup times/costs. **European Journal of Operational Research**, v. 246, n. 2, p. 345–378, 2015. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.ejor.2015.04.004>>. .

ALLAHVERDI, A.; AYDILEK, H. Algorithms for no-wait flowshops with total completion time subject to makespan. **The International Journal of Advanced Manufacturing Technology**, v. 68, n. 9–12, p. 2237–2251, 2013. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84887620001&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

ALLAHVERDI, A.; AYDILEK, H. Total completion time with makespan constraint in no-wait flowshops with setup times. **European Journal of Operational Research**, v. 238, n. 3, p. 724–734, 2014. Elsevier. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84902086496&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

ALLAHVERDI, A.; AYDILEK, H.; AYDILEK, A. No-wait flowshops with two criteria: total completion time and makespan. **European Journal of Operational Research**, v. 269, p. 590–601, 2018.

ALLAHVERDI, A.; AYDILEK, H.; AYDILEK, A. No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan. **Applied Mathematics and Computation**, v. 365, p. 124688, 2020. Elsevier Inc. Disponível em:

<<https://doi.org/10.1016/j.amc.2019.124688>>. .

ALLAHVERDI, A.; GUPTA, J. N. D.; ALDOWAISAN, T. A review of scheduling research involving setup considerations. , v. 27, 1999.

ALLAHVERDI, A.; NG, C. T.; CHENG, T. C. E.; KOVALYOV, M. Y. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research**, v. 187, n. 3, p. 985–1032, 2008. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-36849082395&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

ALLAOUI, H.; ARTIBA, A. Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. **Computers and Industrial Engineering**, v. 47, n. 4, p. 431–450, 2004.

ARMENTANO, V. A.; RONCONI, D. P. Minimização do tempo total de atraso no problema de flowshop com buffer zero através de busca tabu. **Gestão & Produção**, v. 7, n. 3, p. 352–363, 2000. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2000000300011&lng=pt&nrm=iso&tlng=pt>. .

ASHAYERI, J.; TELEN, A.; SELEN, W. A production and maintenance planning model for the process industry. **International Journal of Production Research**, v. 34, n. 12, p. 3311–3326, 1996.

AYDILEK, H.; ALLAHVERDI, A. Heuristics for no-wait flowshops with makespan subject to mean completion time. **Applied Mathematics and Computation**, v. 219, n. 1, p. 351–359, 2012. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84864283085&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

BAO, Y.; JING, H.; ZHENG, L. A hybrid based GA and HS algorithms in blocking flow shop scheduling problems. 2011 2nd International Conference on Mechanic Automation and Control Engineering, MACE 2011 - Proceedings. **Anais...** . p.953–956, 2011. Hohhot: IEEE.

BAO, Y.; ZHENG, L.; JIANG, H. An improved HS algorithms for the blocking flow shop scheduling problems. 2012 International Conference on Computer Science and Information Processing (CSIP). **Anais...** . p.1289–1291, 2012. Shaanxi, China, China: IEEE. Disponível em: <<http://ieeexplore.ieee.org/document/6309097/>>. .

BAUTISTA, J.; CANO, A.; COMPANYS, R.; RIBAS, I. Solving the Fm|block|Cmax problem using Bounded Dynamic Programming. **Engineering Applications of Artificial Intelligence**, v. 25, n. 6, p. 1235–1245, 2012. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84864750347&partnerID=tZOtx3y1>>. .

BOCK, S.; BRISKORN, D.; HORBACH, A. Scheduling flexible maintenance activities subject to job-dependent machine deterioration. **Journal of Scheduling**, v. 15, n. 5, p. 565–578, 2012.

CARAFFA, V.; IANES, S.; BAGCHI, T. P.; et al. Minimizing makespan in a blocking flowshop using genetic algorithms. **International Journal of Production Economics**, v. 70, n. 2, p. 101–115, 2001. Elsevier Science B.V. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-0035277503&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

CARLIER, J.; HAOUARI, M.; KHARBECHÉ, M.; MOUKRIM, A. An optimization-based heuristic for the robotic cell problem. **European Journal of Operational Research**, v. 202, n. 3, p. 636–645, 2010a. Elsevier B.V.

CARLIER, J.; HAOUARI, M.; KHARBECHÉ, M.; MOUKRIM, A. An optimization-based heuristic for the robotic cell problem. **European Journal of Operational Research**, v. 202, n. 3, p. 636–645, 2010b. Elsevier B.V. Disponível em: <<http://dx.doi.org/10.1016/j.ejor.2009.06.035>>. .

CASSADY, C. R.; KUTANOGLU, E. Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. **IEEE Transactions**, v. 35, n. 6, p. 503–513, 2003.

CHAKRAVARTHY, K.; RAJENDRAN, C. A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization. **Production Planning & Control**, v. 10, n. 7, p. 707–714, 1999.

CHEN, H.; ZHOU, S.; LI, X.; XU, R. A hybrid differential evolution algorithm for a two-stage flow shop on batch processing machines with arbitrary release times and blocking. **International Journal of Production Research**, v. 52, n. 19, p. 5714–5734, 2014.

CHEN, J.-S. Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. **European Journal of Operational Research**, v. 190, p. 90–102, 2008.

CHENG, T. C. E.; YANG, S. J.; YANG, D. L. Common due-window assignment and scheduling of linear time-dependent deteriorating jobs and a deteriorating maintenance activity. **International Journal of Production Economics**, v. 135, n. 1, p. 154–161, 2012. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.ijpe.2010.10.005>>. .

CHIU, Y.; SHIH, C. Rescheduling strategies for integrating rush orders with preventive maintenance in a two-machine flow shop. **International Journal of Production Research**, v. 50, n. 20, p. 5783–5794, 2012.

CHOI, B. C.; LEE, K.; LEUNG, J. Y. T.; PINEDO, M. L. Flow shops with machine maintenance: Ordered and proportionate cases. **European Journal of Operational Research**, v. 207, n. 1, p. 97–104, 2010. Elsevier B.V. Disponível em: <<http://dx.doi.org/10.1016/j.ejor.2010.04.018>>. .

CHOWDHURY, A.; GHOSH, A.; SINHA, S. A novel genetic algorithm to solve travelling salesman problem and blocking flow shop scheduling problem. **International Journal Bio-Inspired Computation**, v. 5, n. 5, p. 303–314, 2013.

COHN, A.; KURNAZ, S.; GUAN, Y.; JIANG, Y. Trading off between makespan and customer responsiveness in flow shop sequencing. **International Journal of Production Research**, v. 48, n. 22, p. 6777–6797, 2010.

COMPANYS, R. Métodos heurísticos en la resolución del problema del taller mecánico. **Estudios Empresariales**, v. 5, p. 7–18, 1966.

COMPANYS, R.; MATEO, M. Different behaviour of a double branch-and-bound algorithm on Fm | prmu | Cmax and Fm | block | Cmax problems. **Computers and Operations Research**,

v. 34, n. 4, p. 938–953, 2007.

COMPANYNS, R.; RIBAS, I.; MATEO, MANE. Improvement Tools for NEH Based Heuristics on Permutation and Blocking Flow Shop Scheduling Problems. In: B. Vallespir; T. Alix (Eds.); **Advances in Production Management Systems: New Challenges, New Approaches**. v. 338, p.33–40, 2010.

COMPANYNS, R.; RIBAS, I.; MATEO, MANEL. Note on the behaviour of an improvement heuristic on permutation and blocking flow-shop scheduling. **International Journal of Manufacturing Technology and Management**, v. 20, n. 1/2/3/4, p. 331, 2010.

CROCE, F. DELLA; NARAYAN, V.; TADEI, R. The two-machine total completion time flow shop problem. **European Journal of Operational Research**, v. 90, p. 227–237, 1996.

DANIELS, R. L.; CHAMBERS, R. J. Multi-objective flow shop scheduling. **Naval Research Logistics**, v. 37, p. 981–995, 1990.

DAVENDRA, D.; BIALIC-DAVENDRA, M. Scheduling flow shops with blocking using a discrete self-organising migrating algorithm. **International Journal of Production Research**, v. 51, n. 8, p. 2200–2218, 2013.

DAVENDRA, D.; BIALIC-DAVENDRA, M.; SENKERIK, R.; PLUHACEK, M. Scheduling the flow shop with blocking problem with the chaos-induced discrete self organising migrating algorithm. Proceedings 27th European Conference on Modelling and Simulation. **Anais...**, 2013.

DAVENDRA, D.; SENKERIK, R.; ZELINKA, I.; PLUHACEK, M. Scatter Search Algorithm with chaos based stochasticity. **Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014**, p. 860–866, 2014. IEEE.

DAVENDRA, D.; ZELINKA, I.; BIALIC-DAVENDRA, M.; SENKERIK, R.; JASEK, R. Clustered enhanced differential evolution for the blocking flow shop scheduling problem. **Central European Journal of Operations Research**, v. 20, n. 4, p. 679–717, 2012.

DAVENDRA, D.; ZELINKA, I.; BIALIC-DAVENDRA, M.; SENKERIK, R.; JASEK, R. Discrete Self-Organising Migrating Algorithm for flow-shop scheduling with no-wait makespan. **Mathematical and Computer Modelling**, v. 57, n. 1–2, p. 100–110, 2013. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84866158637&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

DEB, K.; MEMBER, A.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A Fast and Elitist Multiobjective Genetic Algorithm: **IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION**, v. 6, n. 2, p. 182–197, 2002.

DENG, G.; XU, Z.; GU, X. A Discrete Artificial Bee Colony Algorithm for Minimizing the Total Flow Time in the Blocking Flow Shop Scheduling. **Chinese Journal of Chemical Engineering**, v. 20, n. 6, p. 1067–1073, 2012.

DING, J.-Y.; SONG, S.; GUPTA, J. N. D.; et al. New block properties for flowshop scheduling with blocking and their application in an iterated greedy algorithm. **International Journal of Production Research**, v. 7543, n. September, p. 1–14, 2015. Disponível em: <<http://www.tandfonline.com/doi/full/10.1080/00207543.2015.1076941>>. .

DUAN, J. H.; PAN, Q. K.; GAO, K. Z.; LI, J.; BAO, Z. Scheduling the blocking flow shop problem using a harmony search algorithm. *Proceedings - 2010 6th International Conference on Natural Computation, ICNC 2010. Anais...* v. 8, p.4263–4266, 2010. IEEE.

DUTTA, S. K.; CUNNINGHAM, A. A. Sequencing Two-Machine Flow-Shops with Finite Intermediate Storage. **Management Science**, v. 21, n. 9, p. 989–996, 1975.

EDDALY, M.; JARBOUI, B.; SIARRY, P. Combinatorial Particle Swarm Optimization for solving Blocking Flowshop Scheduling Problem. **Journal of Computational Design and Engineering**, v. 3, n. 4, p. 295–311, 2016. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.jcde.2016.05.001>>. .

EMMONS, H.; VAIRAKTARAKIS, G. **Flow Shop Scheduling Theoretical Results, Algorithms, and Applications**. New York: Springer, 2013a.

EMMONS, H.; VAIRAKTARAKIS, G. Blocking or Limited Buffers in Flow Shops. **Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications**. p.221–245, 2013b. Boston, MA: Springer US. Disponível em: <https://doi.org/10.1007/978-1-4614-5152-5_7>. .

FEO, T. A.; RESENDE, M. G. C. R. A probabilistic heuristic for a computationally difficult set covering problem. **Operations Research Letters**, v. 8, n. April, p. 67–71, 1989.

FERNANDEZ-VIAGAS, V.; LEISTEN, R.; FRAMINAN, J. M. A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. **Expert Systems With Applications**, v. 61, p. 290–301, 2016. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.eswa.2016.05.040>>. .

FINK, A.; VOSS, S. Solving the continuous flow-shop scheduling problem by metaheuristics. **European Journal of Operational Research**, v. 151, n. 2, p. 400–414, 2003.

FOLEY, R. D.; SURESH, S. Minimizing the expected flowtime in stochastic flow shops. **IIE Transactions (Institute of Industrial Engineers)**, v. 16, n. 4, p. 391–395, 1984.

FRAMINAN, J. M.; LEISTEN, R. A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness. **International Journal of Production Economics**, v. 99, n. 1–2, p. 28–40, 2006.

FU, Y.; WANG, H.; TIAN, G.; LI, Z.; HU, H. Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm. **Journal of Intelligent Manufacturing**, , n. 2014, 2018. Springer US.

FU, Y.; WANG, Z. Z.; ZHANG, J.; WANG, Z. Z. A blocking flow shop deteriorating scheduling problem via a hybrid chemical reaction optimization. **Advances in Mechanical Engineering**, v. 9, n. 6, p. 1–10, 2017.

GENDREAU, M.; HERTZ, A.; LAPORTE, G. New insertion and postoptimization procedures for the travelling salesman problem. **Operations Research**, v. 40, n. 6, p. 1086–1094, 1992.

GILMORE, P. C.; GOMORY, R. E. Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem. **Operations Research**, 1964.

GONG, D.; HAN, Y.; SUN, J. A novel hybrid multi-objective artificial bee colony algorithm

for blocking lot-streaming flow shop scheduling problems. **Knowledge-Based Systems**, v. 148, p. 115–130, 2018. Elsevier B.V. Disponível em: <<https://doi.org/10.1016/j.knosys.2018.02.029>>. .

GONG, H.; TANG, L. Two-machine flowshop scheduling problem with transportation and blocking features. 2008 Chinese Control and Decision Conference (CCDC 2008). **Anais...** . p.1471–1474, 2008. Yantai, China: IEEE.

GONG, H.; TANG, L.; DUIN, C. W. A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. **Computers & Operations Research**, v. 37, n. 5, p. 960–969, 2010. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2009.08.001>>. .

GRABOWSKI, J.; PEMPERA, J. Sequencing of jobs in some production system. **European Journal of Operational Research**, v. 125, n. 3, p. 535–550, 2000. Elsevier Science B.V. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-0034274968&partnerID=tZOtx3y1>>. Acesso em: 14/11/2014.

GRABOWSKI, J.; PEMPERA, J. The permutation flow shop problem with blocking. A tabu search approach. **Omega-International Journal of Management Science**, v. 35, n. 3, p. 302–311, 2007.

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; KAN, A. H. G. R. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. **Annals of Discrete Mathematics**, 1979.

GRAVES, G. H.; LEE, C.-Y. Scheduling maintenance and semiresumable jobs on single machines. **Naval Research Logistics**, v. 46, n. 7, p. 845–863, 1999.

GUANLONG, D.; SHUNING, Z.; MEI, Z. A discrete group search optimizer for blocking flow shop multi-objective scheduling. **Advances in Mechanical Engineering**, v. 8, n. 8, p. 1–9, 2016.

HADDA, H.; LAMOURI, S.; ARTIBA, A.; AGHEZZAF, E. Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. **International Journal of Production Economics**, v. 112, p. 161–167, 2008.

HALL, N. G. .; SRISKANDARAJAH, C. A survey of machine scheduling problems with blocking and no-wait in process. **Operations Research**, v. 44, n. 3, p. 510–525, 1996. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-0030134958&partnerID=tZOtx3y1>>. .

HAN, Y.-Y.; DUAN, J.-H.; ZHANG, M.; IEEE. Apply the discrete artificial bee colony algorithm to the blocking flow shop problem with makespan criterion. **2011 Chinese Control and Decision Conference, Vols 1-6**. p.2131–2135, 2011.

HAN, Y.-Y.; GONG, D.; SUN, X. A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. **Engineering Optimization**, v. 47, n. 7, p. 927–946, 2015.

HAN, Y.-Y.; PAN, Q.-K.; LI, J.-Q.; SANG, H. An improved artificial bee colony algorithm for the blocking flowshop scheduling problem. **The International Journal of Advanced**

Manufacturing Technology, v. 60, n. 9–12, p. 1149–1159, 2012. Disponível em: <<http://link.springer.com/10.1007/s00170-011-3680-0>>. .

HAN, Y.; GONG, D.; JIN, Y.; PAN, Q. Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time. **Applied Soft Computing Journal**, v. 42, p. 229–245, 2016. Elsevier B.V. Disponível em: <<http://dx.doi.org/10.1016/j.asoc.2016.01.033>>. .

HAN, Y.; GONG, D.; JIN, Y.; PAN, Q. Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. **IEEE Transactions on Cybernetics**, v. 49, n. 1, p. 184–197, 2019. IEEE.

HAN, Y.; GONG, D.; LI, J.; ZHANG, Y. Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. **International Journal of Production Research**, v. 54, n. May, p. 6782–6797, 2016.

HAN, Y.; LI, J.; GONG, D.; SANG, H. Multi-Objective Migrating Birds Optimization Algorithm for Stochastic Lot-Streaming Flow Shop Scheduling With Blocking. **IEEE Access**, v. 7, p. 5946–5962, 2019. IEEE.

HAN, Y. Y.; LIANG, J. J.; PAN, Q. K.; et al. Effective hybrid discrete artificial bee colony algorithms for the total flowtime minimization in the blocking flowshop problem. **International Journal of Advanced Manufacturing Technology**, v. 67, n. 1–4, p. 397–414, 2013.

HNAIEN, F.; YALAOUI, F. **A bi-criteria flow-shop scheduling with preventive maintenance**. IFAC, 2013.

HU, R.; WANG, L.; QIAN, B.; HUANG, F. Z. Differential evolution method for stochastic flow shop scheduling with limited buffers. **2008 IEEE Congress on Evolutionary Computation, CEC 2008**, v. 7, p. 1295–1301, 2008.

ISHIBUCHI, H.; YOSHIDA, T.; MURATA, T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. **IEEE Transactions on Evolutionary Computation**, v. 7, n. 2, p. 204–223, 2003.

JANUARIO, T. DE O.; ARROYO, J. E. C.; MOREIRA, M. C. O. Genetic Algorithm for Tardiness Minimization in Flowshop with Blocking. In: N. Krasnogor; M. B. Melián-Batista; J. A. M. Pérez; J. M. Moreno-Vega; D. A. Pelta (Eds.); **Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)**. p.153–164, 2009. Berlin, Heidelberg, Heidelberg: Springer Berlin Heidelberg. Disponível em: <https://doi.org/10.1007/978-3-642-03211-0_13>. .

JARBOUI, B.; EDDALY, M.; SIARRY, P.; REBAÏ, A. An Estimation of Distribution Algorithm for Minimizing the Makespan in Blocking Flowshop Scheduling Problems. In: U. K. Chakraborty (Ed.); **Computational Intelligence in Flow Shop and Job Shop Scheduling**. p.151–167, 2009. Berlin, Heidelberg: Springer Berlin Heidelberg.

JIA, C. Minimizing variation in stochastic flow shop. **Operations Research Letters**, v. 23, n. 3–5, p. 109–111, 1998.

KALCZYNSKI, P. J. A. N.; KAMBUROWSKI, J. Two-machine stochastic flow shops with

blocking and the travelling salesman problem. **Journal of Scheduling**, v. 8, n. 1993, p. 529–536, 2005.

KAMOUN, H.; SRISKANDARAJAH, C. The complexity of scheduling jobs in repetitive manufacturing systems. **European Journal of Operational Research**, v. 70, n. 3, p. 350–364, 1993. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-0027906646&partnerID=tZOtx3y1>>. .

KARABOGA, D.; BASTURK, B. On the performance of artificial bee colony (ABC) algorithm. **Applied Soft Computing**, v. 8, p. 687–697, 2008.

KARABULUT, K. A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. **Computers and Industrial Engineering**, v. 98, p. 300–307, 2016. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2016.06.012>>. .

KELLEGÖZ, T.; TOKLU, B.; WILSON, J. Elite guided steady-state genetic algorithm for minimizing total tardiness in flowshops. **Computers and Industrial Engineering**, v. 58, n. 2, p. 300–306, 2010. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2009.11.001>>. .

KHAMSEH, A.; JOLAI, F.; BABAEI, M. Integrating sequence-dependent group scheduling problem and preventive maintenance in flexible flow shops. **International Journal of Advanced Manufacturing Technology**, v. 77, n. 1–4, p. 173–185, 2015.

KHELIFATI, S. L.; BENBOUZID-SITAYEB, F. A multi-agent scheduling approach for the joint scheduling of jobs and maintenance operations in the flow shop sequencing problem. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 6923 LNAI, n. PART 2, p. 60–69, 2011.

KHORASANIAN, D.; MOSLEHI, G. An Iterated Greedy Algorithm for solving the blocking flow shop scheduling problem with total flow time criteria. **International Journal of Industrial Engineering & Production Research**, v. 23, n. 4, p. 301–308, 2012.

KHORASANIAN, D.; MOSLEHI, G. Two-machine flow shop scheduling problem with blocking, multi-task flexibility of the first machine, and preemption. **Computers and Operations Research**, v. 79, n. September 2016, p. 94–108, 2017. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2016.09.023>>. .

KHORRAMIZADEH, M.; RIAHI, V. A Bee Colony Optimization Approach for Mixed Blocking Constraints Flow Shop Scheduling Problems. **Mathematical Problems in Engineering**, v. 2015, p. 1–10, 2015.

KOULAMAS, C.; PANWALKAR, S. S. New index priority rules for no-wait flow shops. **Computers and Industrial Engineering**, v. 115, n. November 2017, p. 647–652, 2018. Elsevier. Disponível em: <<https://doi.org/10.1016/j.cie.2017.12.015>>. .

LAGEWEG, B. J.; LENSTRA, J. . K.; KAN, A. . H. . G. . R. A General Bounding Scheme for the Permutation Flow-Shop Problem. **Operations Research**, v. 26, n. 1, p. 53–67, 1978.

LEBBAR, G.; ABBASSI, A. EL; JABRI, A.; BARKANI, A. EL; DARCHERIF, M. Multi - criteria blocking flow shop scheduling problems : Formulation and performance analysis.

Advances in Production Engineering & Management, v. 13, n. 2, p. 136–146, 2018.

LEBBAR, G.; BARKANY, A. E.; IKRAM, E. A.; MOUMEN, D. A. Solving the Multi Objective Flow Shop Scheduling Problems Using an Improved NSGA-II. **International Journal of Operations and Quantitative Management**, v. 24, n. 3, p. 211–230, 2018.

LEE, C.-Y.; CHEN, Z.-L. Scheduling jobs and maintenance activities on parallel machines. **Naval Research Logistics**, v. 47, p. 145–165, 2000.

LEE, C. Y.; LIN, C. S. Single-machine scheduling with maintenance and repair rate-modifying activities. **European Journal of Operational Research**, v. 135, n. 3, p. 493–513, 2001.

LEE, W. C.; SHIAU, Y. R.; CHEN, S. K.; WU, C. C. A two-machine flowshop scheduling problem with deteriorating jobs and blocking. **International Journal of Production Economics**, v. 124, n. 1, p. 188–197, 2010. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.ijpe.2009.11.001>>. .

LEINSTEN, R. Flowshop sequencing problems with limited buffer storage. **International Journal of Production Research**, v. 28, n. 11, p. 2085–2100, 1990.

LEVNER, E.; KOGAN, K.; MAIMON, O. Flowshop scheduling of robotic cells with job-dependent transportation and set-up effects. **Journal of the Operational Research Society**, v. 46, p. 1447–1455, 1995.

LEVNER, E. V. Optimal planning of parts machining on a number of machines. **Automatic Remote Control**, v. 12, p. 1972–1978, 1969.

LI, S. H.; TANG, L. X. A tabu search algorithm based on new block properties and speed-up method for permutation flow-shop with finite intermediate storage. **Journal of Intelligent Manufacturing**, v. 16, n. 4–5, p. 463–477, 2005.

LI, Y.; WANG, S. ; DING, L.; XIE, X. A dynamic programming based heuristic in Max-algebra for solving a blocking flow-shop problem. Proceedings of 2013 2nd International Conference on Measurement, Information and Control. **Anais...** . p.375–379, 2013. IEEE. Disponível em: <<http://ieeexplore.ieee.org/document/6757986/>>. .

LIANG, J. J.; PAN, Q.-K.; CHEN, T.; WANG, L. Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer. **International Journal of Advanced Manufacturing Technology**, v. 55, n. 5–8, p. 755–762, 2011.

LIN, S. W.; YING, K. C. Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. **Omega (United Kingdom)**, v. 41, n. 2, p. 383–389, 2013. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.omega.2012.03.006>>. .

LIN, S.; YING, K. Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm. **Computers & Operations Research**, v. 40, n. 6, p. 1625–1647, 2013. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2011.08.009>>. .

LIU, B.; WANG, L.; JIN, Y.-H. An effective hybrid particle swarm optimization for no-wait flow shop scheduling. **International Journal of Advanced Manufacturing Technology**, v. 31, n. 9–10, p. 1001–1011, 2007. Disponível em:

<<http://www.scopus.com/inward/record.url?eid=2-s2.0-33846194668&partnerID=tZOtx3y1>>. Acesso em: 14/11/2014.

LIU, B.; WANG, L.; JIN, Y. H. An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. **Computers & Operations Research**, v. 35, n. 9, p. 2791–2806, 2008.

LIU, B.; WANG, L.; QIAN, B.; JIN, Y. Hybrid particle swarm optimization for stochastic flow shop scheduling with no-wait constraint. IFAC Proceedings Volumes (IFAC-PapersOnline). **Anais...** v. 17, 2008. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-79961017861&partnerID=tZOtx3y1>>. .

LIU, J.; REEVES, C. R. Constructive and composite heuristic solutions to the P // C i scheduling problem. **European Journal of Operational Research**, v. 132, n. 2, p. 439–452, 2001. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0377221700001375>>. .

LIU, S. Q.; KOZAN, E. Scheduling a flow shop with combined buffer conditions. **International Journal of Production Economics**, v. 117, n. 2, p. 371–380, 2009. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-58949084435&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

LIU, S.; WANG, P.; ZHANG, J. An improved biogeography-based optimization algorithm for blocking flow shop scheduling problem. **Chinese Journal of Electronics**, v. 27, n. 2, p. 351–358, 2018.

LIU, X.; LI, K.; REN, H. A Hybrid Algorithm for the Permutation Flowshop Scheduling Problem without Intermediate Buffers. **Discrete Dinyamics in Nature and Society**, v. 2015, 2015.

LOGENDRAN, R.; SRISKANDARAJAH, C. Two-machine group scheduling problem with blocking and anticipatory setups. **European Journal of Operational Research**, v. 69, n. 3, p. 467–481, 1993.

LUO, W.; CHENG, T. C. E.; JI, M. Single-machine scheduling with a variable maintenance activity. **Computers and Industrial Engineering**, v. 79, p. 168–174, 2015. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2014.11.002>>. .

LUO, W.; JI, M. Scheduling a variable maintenance and linear deteriorating jobs on a single machine. **Information Processing Letters**, v. 115, n. 1, p. 33–39, 2015. Elsevier B.V. Disponível em: <<http://dx.doi.org/10.1016/j.ipl.2014.08.011>>. .

MA, Y.; CHU, C.; ZUO, C. A survey of scheduling with deterministic machine availability constraints. **Computers and Industrial Engineering**, v. 58, n. 2, p. 199–211, 2010. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2009.04.014>>. .

MACCARTHY, B. L.; LIU, J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research**, v. 31, n. 1, p. 59–79, 1993. Disponível em: <<http://dx.doi.org/10.1080/00207549308956713>>. .

MAHDAVI, M.; FESANGHARY, M.; DAMANGIR, E. An improved harmony search

algorithm for solving optimization problems. **Applied Mathematics and Computation**, v. 188, p. 1567–1579, 2007.

MALEKI-DAROUNKOLAEI, A.; MODIRI, M.; TAVAKKOLI-MOGHADDAM, R.; SEYYEDI, I. A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. **Journal of Industrial Engineering International**, v. 8, n. 26, p. 603–622, 2012.

MANJESHWAR, P. K.; DAMODARAN, P.; SRIHARI, K. Minimizing makespan in a flow shop with two batch-processing machines using simulated annealing. **Robotics and Computer-Integrated Manufacturing**, v. 25, p. 667–679, 2009.

MANJESHWAR, P. K.; DAMODARAN, P.; SRIHARI, K. Genetic algorithms for minimizing makespan in a flow shop with two capacitated batch processing machines. **International Journal of Advanced Manufacturing Technology**, v. 55, p. 1171–1182, 2011.

MARTINEZ, S. **Ordonnancement de systèmes de production avec contraintes de blocage**, 2005. IRCCyN, Nntes, France.

MARTINEZ, S.; DAUZÈRE-PÉRÈS, S.; GUÉRET, C.; MATI, Y.; SAUER, N. Complexity of flowshop scheduling problems with a new blocking constraint. **European Journal of Operational Research**, v. 169, n. 3, p. 855–864, 2006.

MARTINEZ, S.; GUÉRET, C.; SAUER, N. A simulated annealing method for the flow-shop with a new blocking constraint. 20th European conference on operational research (EURO'04). **Anais...** . p.27, 2004. Rhodes, Greece, Greece.

MCCORMICK, S. T.; PINEDO, M. L.; SHENKER, S.; WOLF, B. Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. **Operations Research**, v. 37, n. 6, p. 925–935, 1989.

MCCORMICK, S. T.; PINEDO, M. L.; WOLF, B. **The complexity of scheduling a flexible assembly line with finite buffers**. New York, 1987.

MIYATA, H. H.; NAGANO, M. S. The blocking flow shop scheduling problem: A comprehensive and conceptual review. **Expert Systems with Applications**, v. 137, p. 130–156, 2019. Elsevier Ltd.

MIYATA, H. H.; NAGANO, M. S.; GUPTA, JATINDER N.D. Incorporating preventive maintenance into the m-machine no-wait flow-shop scheduling problem with total flow-time minimization: a computational study. **Engineering Optimization**, v. 51, n. 4, p. 680–698, 2019. Disponível em: <<https://doi.org/10.1080/0305215X.2018.1485903>>. .

MIYATA, H. H.; NAGANO, M. S.; GUPTA, JATINDER N D. Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. **Computers & Industrial Engineering**, v. 135, n. June, p. 79–104, 2019. Elsevier.

MIYATA, H. H.; NAGANO, M. S.; GUPTA, J. N. D. Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. **Computers & Industrial Engineering**, v. 135, n. June, p. 79–104, 2019. Elsevier. Disponível em: <<https://doi.org/10.1016/j.cie.2019.05.034>>. .

MLADENOVIC, N.; HANSEN, P. Variable Neighborhood Search. **Computers & Operations Research**, v. 24, n. 1, p. 1097–1100, 1997.

MOR, B.; MOSHEIOV, G. Scheduling a deteriorating maintenance activity and due-window assignment based on common flow allowance. **International Journal of Production Economics**, v. 135, p. 222–230, 2012. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.ijpe.2011.07.013>>. .

MOSHEIOV, G.; SARIG, A. Scheduling a maintenance activity to minimize total weighted completion-time. **Computers and Mathematics with Applications**, v. 57, n. 4, p. 619–623, 2009a. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.camwa.2008.11.008>>. .

MOSHEIOV, G.; SARIG, A. Scheduling a maintenance activity and due-window assignment on a single machine. **Computers and Operations Research**, v. 36, n. 9, p. 2541–2545, 2009b.

MOSLEHI, G.; KHORASANIAN, D. Optimizing blocking flow shop scheduling problem with total completion time criterion. **Computers & Operations Research**, v. 40, n. 7, p. 1874–1883, 2013.

MOSLEHI, G.; KHORASANIAN, D. A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion. **Computers and Operations Research**, v. 52, p. 260–268, 2014. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2013.09.014>>. .

NAGANO, M. S.; KOMESU, A. S.; MIYATA, H. H. An evolutionary clustering search for the total tardiness blocking flow shop problem. **Journal of Intelligent Manufacturing**, 2017. Springer US.

NAGANO, M. S.; MOCCELLIN, J. V. A high quality solution constructive heuristic for flow shop sequencing. **Journal of the Operational Research Society**, v. 53, p. 1374–1379, 2002.

NAGANO, M. S.; SILVA, A. A. DA; LORENA, L. A. N. A new evolutionary clustering search for a no-wait flow shop problem with set-up times. **Engineering Applications of Artificial Intelligence**, v. 25, n. 6, p. 1114–1120, 2012. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84864774170&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

NAWAZ, M.; ENSCORE JR, E. E.; HAM, I. A Heuristic Algorithm for the m-Machine , n-Job Flow-shop Sequencing Problem. **Omega-International Journal of Management Science**, v. 11, n. 1, p. 91–95, 1983.

NORMAN, B. A. Scheduling flowshops with finite buffers and sequence-dependent setup times. **Computers and Industrial Engineering**, v. 36, n. 1, p. 163–177, 1999.

NOUHA, N.; TALEL, L. A Particle Swarm Optimization Metaheuristic for the Blocking Flow Shop Scheduling Problem: Total Tardiness Minimization. In: M. Rovatsos; G. Vouras; V. Julian (Eds.); Multi-Agent Systems and Agreement Technologies. **Anais...** . p.145–153, 2016. Cham: Springer International Publishing.

NOURI, N.; LADHARI, T. Minimizing Regular Objectives for Blocking Permutation Flow Shop Scheduling: Heuristic Approaches. GECCO '15 Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. **Anais...** . p.441–448, 2015. Madrid,

Spain: ACM New York.

NOURI, N.; LADHARI, T. Evolutionary multiobjective optimization for the multi-machine flow shop scheduling problem under blocking. **Annals of Operations Research**, p. 1–18, 2017. Springer US.

NOWICKI, E. The permutation flow shop with buffers : A tabu search approach. **European Journal of Industrial Engineering**, v. 116, p. 205–219, 1999.

OGBU, F. A.; SMITH, D. K. THE APPLICATION OF THE SIMULATED ANNEALING ALGORITHM TO THE SOLUTION THE $n/m/C_{max}$ FLOWSHOP PROBLEM. **Computers & Operations Research**, v. 17, n. 3, p. 243–253, 1990.

OMRAN, M. G. H. Global-best harmony search. **Applied Mathematics and Computation**, v. 198, p. 643–656, 2008.

OSMAN, I. H.; LAPORTE, G. Metaheuristics: A bibliography. **Annals of Operations Research**, v. 63, n. 5, p. 511–623, 1996.

OSMAN, I.; POTTS, C. N. Simulated Annealing for Permutation Flow-Shop Scheduling. **Omega-International Journal of Management Science**, v. 17, n. 6, p. 551–557, 1989.

OZOLINS, A. Improved bounded dynamic programming algorithm for solving the blocking flow shop problem. **Central European Journal of Operations Research**, v. 27, n. 1, p. 15–38, 2019. Springer Berlin Heidelberg.

PAN, Q.-K.; WANG, L.; GAO, L. A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers. **Applied Soft Computing**, v. 11, n. 8, p. 5270–5280, 2011. Elsevier B.V. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1568494611001931>%5Cnhttp://www.sciencedirect.com/science?_ob=ImageURL&_cid=272229&_user=12009445&_pii=S1568494611001931&_check=y&_origin=article&_zone=toolbar&_coverDate=31-Dec-2011&view=c&originConte>. .

PAN, Q.; FATIH, M.; LIANG, Y. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. **Computers & Industrial Engineering**, v. 55, n. 4, p. 61–82, 2008. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2008.03.003>>. .

PAN, Q. K.; WANG, L. Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. **Omega**, v. 40, n. 2, p. 218–229, 2012. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.omega.2011.06.002>>. .

PAN, Q. K.; WANG, L.; GAO, L.; LI, W. D. An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers. **Information Sciences**, v. 181, n. 3, p. 668–685, 2011.

PAN, Q.; WANG, L.; SANG, H.; LI, J.; LIU, M. A High Performing Memetic Algorithm for the Flowshop Scheduling Problem With Blocking. **IEEE Transactions on Automation Science and Engineering**, v. 10, n. 3, p. 741–756, 2013. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6353603>>. .

PANWALKAR, S. S.; KOULAMAS, C. The evolution of schematic representations of flow

shop scheduling problems. **Journal of Scheduling**, 2018. Springer US. Disponível em: <<https://doi.org/10.1007/s10951-018-0594-9>>. .

PAPADIMITRIOU, C. H.; KANELLAKIS, P. C. Flowshop scheduling with limited temporary storage. **Journal of the ACM**, v. 27, n. 3, p. 533–549, 1980. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-0018210671&partnerID=tZOtx3y1>>. .

PINEDO, M. L. Minimizing Expected Makespan in Stochastic Flow Shops. **Operations Research**, v. 30, n. 1, p. 148–162, 1982. Disponível em: <<http://www.jstor.org/stable/1427030>>. .

PINEDO, M. L. **Scheduling Theory, Algorithms, and Systems**. 3rd ed. Nova York: Prentice Hall, 2008.

PITTY, S. S.; KARIMI, I. A. Novel MILP models for scheduling permutation flowshops. **Chemical Product and Process Modeling**, v. 3, n. 1, 2008.

PONNAMBALAM, S. G. ; JAWAHAR, N. ;; CHANDRASEKARAN, S. Discrete Particle Swarm Optimization Algorithm for Flowshop Scheduling. In: A. Lazinica (Ed.); **Particle Swarm Optimization**, 2009. InTech. Disponível em: <http://www.intechopen.com/books/particle_swarm_optimization/discrete_particle_swarm_optimization_algorithm%0Afor_flowshop_scheduling>. .

POUR, H. D. A new heuristic for the n -job , m -machine ow-shop problem. **Production Planning & Control**, v. 12, n. January, p. 648–653, 2001.

PRASAD, S. D.; RAJENDRAN, C.; CHETTY, O. V. K. A genetic algorithmic approach to multi-objective scheduling in a Kanban-controlled flowshop with intermediate buffer and transport constraints. **International Journal of Advanced Manufacturing Technology**, v. 29, n. 5, p. 564–576, 2006.

QI, X.; CHEN, T.; TU, F. Scheduling the Maintenance on a Single Machine. **Journal of the Operational Research Society**, v. 50, n. 10, p. 1071–1078, 1999.

QIAN, BIN; WANG, L.; HUANG, D.; WANG, W.; WANG, X. An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. **Computers & Operations Research**, v. 36, p. 209–233, 2009.

QIAN, B.; WANG, L.; HUANG, D. X.; WANG, X. An effective hybrid DE-based algorithm for flow shop scheduling with limited buffers. **International Journal of Production Research**, v. 47, n. 1, p. 1–24, 2009.

RAJENDRAN, C.; ZIEGLER, H. An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. **European Journal of Operational Research**, v. 103, n. 1, p. 129–138, 1997.

RAMESHKUMAR, K.; SURESH, R.K.; MOHANASUNDARAM, K. M. Discrete Particle Swarm Optimization (DPSO) Algorithm for Permutation Flowshop Scheduling to Minimize Makespan. Advances in Natural Computation. ICNC 2005. Lecture Notes in Computer Science. **Anais...**, 2005. Berlin: Springer, Berling, Heidelberg.

RAZA, S. A.; AL-TURKI, U. M.; SELIM, S. Z. Early-Tardy Minimization for Joint Scheduling

of Jobs and Maintenance Operations on a Single Machine. **International Journal of Operations Research**, v. 4, n. 1, p. 32–41, 2007.

REBAI, M.; KACEM, I.; ADJALLAH, K. H. Scheduling jobs and maintenance activities on parallel machines. **Operational Research**, v. 13, n. 3, 2013.

REDDI, S. S.; RAMAMOORTHY, C. V. On the flow-shop sequencing with no wait in process. **Operations Research Quarterly**, v. 23, n. 3, p. 323–328, 1972.

REN, T.; LIANG, J.; LIU, Y. Y.; et al. Blocking Flowshop Scheduling Model for Optimising Cubic Criterion with Release Dates. **IEEE Access**, v. 6, n. 2011, p. 72179–72188, 2018.

REZA HEJAZI, S.; SAGHAFIAN, S. Flowshop-scheduling problems with makespan criterion: a review. **International Journal of Production Research**, v. 43, n. 14, p. 2895–2929, 2005.

RIAHI, V.; KHORRAMIZADEH, M.; HAKIM NEWTON, M. A.; SATTAR, A. Scatter search for mixed blocking flowshop scheduling. **Expert Systems with Applications**, v. 79, p. 20–32, 2017. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.eswa.2017.02.027>>. .

RIAHI, V.; NEWTON, M. A. H.; SU, K.; SATTAR, A. Local search for flowshops with setup times and blocking constraints. Proceedings International Conference on Automated Planning and Scheduling, ICAPS. **Anais...** . v. 2018-June, p.199–207, 2018.

RIBAS, I.; COMPANYS, R. A competitive variable neighbourhood search algorithm for the blocking flow shop problem. **European Journal of Industrial Engineering**, v. 7, n. 6, p. 729–754, 2013.

RIBAS, I.; COMPANYS, R. Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. **Computers and Industrial Engineering**, v. 87, p. 30–39, 2015. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2015.04.013>>. .

RIBAS, I.; COMPANYS, R.; TORT-MARTORELL, X. An iterated greedy algorithm for the flowshop scheduling problem with blocking. **Omega**, v. 39, n. 3, p. 293–301, 2011. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.omega.2010.07.007>>. .

RIBAS, I.; COMPANYS, R.; TORT-MARTORELL, X. An efficient iterated local search algorithm for the total tardiness blocking flow shop problem. **International Journal of Production Research**, v. 51, n. 17, p. 5238–5252, 2013.

RIBAS, I.; COMPANYS, R.; TORT-MARTORELL, X. An efficient Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with total flowtime minimization. **Expert Systems with Applications**, v. 42, n. 15–16, p. 37–41, 2015.

RONCONI, D. P. A note on constructive heuristics for the flowshop problem with blocking. **International Journal of Production Economics**, v. 87, n. 1, p. 39–48, 2004.

RONCONI, D. P. A Branch-and-Bound Algorithm to Minimize the Makespan in a Flowshop with Blocking. **Annals of Operations Research**, v. 138, p. 53–65, 2005.

RONCONI, D. P.; ARMENTANO, V. A. Lower bounding schemes for flowshops with blocking in-process. **Journal of the Operational Research Society**, 2001.

RONCONI, D. P.; HENRIQUES, L. R. S. Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. **Omega**, v. 37, n. 2, p. 272–281, 2009.

ROSSIT, D. A.; TOHMÉ, F.; FRUTOS, M. The Non-Permutation Flow-Shop scheduling problem : A literature review R. **Omega**, v. 77, p. 143–153, 2018. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.omega.2017.05.010>>. .

RUIZ, R.; GARCÍA-DÍAZ, J. C.; MAROTO, C. Considering scheduling and preventive maintenance in the flowshop sequencing problem. **Computers and Operations Research**, v. 34, n. 11, p. 3314–3330, 2007.

RUIZ, R.; MAROTO, C.; ALCARAZ, J. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. **European Journal of Operational Research**, v. 165, n. 1, p. 34–54, 2005.

RUIZ, R.; STÜTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, v. 177, n. 3, p. 2033–2049, 2007.

RUIZ, R.; STÜTZLE, T. An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. **European Journal of Operational Research**, v. 187, n. 3, p. 1143–1159, 2008.

RUIZ, R.; VÁZQUEZ-RODRÍGUEZ, J. A. The hybrid flow shop scheduling problem. **European Journal of Operational Research**, v. 205, n. 1, p. 1–18, 2010. Elsevier B.V. Disponível em: <<http://dx.doi.org/10.1016/j.ejor.2009.09.024>>. .

SADAQA, M.; MORAGA, R. J. Scheduling Blocking Flow Shops Using Meta-RaPS. **Procedia Computer Science**, v. 61, p. 533–538, 2015. Elsevier Masson SAS. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1877050915030410>>. .

SAFARI, E.; SADJADI, S. J.; SHAHANAGHI, K. Scheduling flowshops with condition-based maintenance constraint to minimize expected makespan. **International Journal of Advanced Manufacturing Technology**, v. 46, n. 5–8, p. 757–767, 2010.

SAUER, C.; SAUVEY, N. A genetic algorithm with genes-association recognition for flowshop scheduling problems. **Journal of Intelligent Manufacturing**, v. 23, p. 1167–1177, 2012.

SAUVEY, C.; SAUER, N. Heuristique pour la résolution des problèmes d’ordonnancement de type flowshop avec blocage RCb. 10ème conférence de la Société Française de Recherche Opérationnelle et d’Aide à la Décision (ROADEF’09). **Anais...** . p.236–237, 2009. Nancy, France, France.

SHAO, Z.; PI, D.; SHAO, W. Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. **Computers & Industrial Engineering**, v. 111, p. 331–351, 2017. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2017.07.037>>. .

SHAO, ZHONGSHI; PI, D.; SHAO, W. A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. **Swarm and Evolutionary Computation**, v. 40, n. December 2017, p. 53–75, 2018a. Elsevier Ltd.

SHAO, Z.; PI, D.; SHAO, W. Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem. **Engineering Optimization**, v. 50, n. 5, p. 894–916, 2018. Taylor & Francis. Disponível em: <<https://doi.org/0305215X.2017.1353090>>. .

SHAO, ZHONGSHI; PI, D.; SHAO, W. A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. **Swarm and Evolutionary Computation**, v. 40, n. December 2017, p. 53–75, 2018b. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.swevo.2017.12.005>>. .

SHAO, ZHONGSHI; PI, D.; SHAO, W. A multi-objective discrete invasive weed optimization for multi-objective blocking flow-shop scheduling problem. **Expert Systems With Applications**, v. 113, p. 77–99, 2018c. Elsevier Ltd. Disponível em: <<https://doi.org/10.1016/j.eswa.2018.06.020>>. .

SHEN, J. N.; WANG, L.; WANG, S. Y. A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. **Knowledge-Based Systems**, v. 74, p. 167–175, 2015. Elsevier B.V. Disponível em: <<http://dx.doi.org/10.1016/j.knosys.2014.11.016>>. .

SHEN, X.; HAN, Y.; FU, J. Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. **Soft Computing**, v. 21, n. 21, p. 6531–6554, 2017. Springer Berlin Heidelberg.

SITAYEB, F. B.; GUEBLI, S. A.; BESSADI, Y.; VARNIER, C.; ZERHOUNI, N. Joint scheduling of jobs and Preventive Maintenance operations in the flowshop sequencing problem: a resolution with sequential and integrated strategies. **International Journal of Manufacturing Research**, v. 6, n. 1, p. 30–48, 2011.

SORTRAKUL, N.; NACHTMANN, H. L.; CASSADY, C. R. Genetic algorithms for integrated preventive maintenance planning and production scheduling for a single machine. **Computers in Industry**, v. 56, n. 2, p. 161–168, 2005.

SOUKHAL, A.; OULAMARA, A.; MARTINEAU, P. Complexity of flow shop scheduling problems with transportation constraints. **European Journal of Operational Research**, v. 161, n. 1, p. 32–41, 2005.

SUHAMI, I.; MAH, R. S. H. An implicit enumeration scheme for the flowshop problem with no intermediate storage. **Computers and Chemical Engineering**, v. 5, n. 2, p. 83–91, 1981.

SUN, K.; LI, H. Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines. **International Journal of Production Economics**, v. 124, n. 1, p. 151–158, 2010. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.ijpe.2009.10.018>>. .

TAILLARD, E. Benchmarks for Basic Scheduling Problems. **European Journal of Operational Research**, v. 64, p. 278–285, 1993.

TAKANO, M. I.; NAGANO, M. S. A branch-and-bound method to minimize the makespan in a permutation flow shop with blocking and setup times. **Cogent Engineering**, v. 4, n. 1, p. 1–16, 2017. Cogent. Disponível em: <<http://doi.org/10.1080/23311916.2017.1389638>>. .

TAKANO, M. I.; NAGANO, M. S. Evaluating the performance of constructive heuristics for

the blocking flow shop scheduling problem with setup times. **International Journal of Industrial Engineering Computations**, v. 10, n. 1, p. 37–50, 2019.

TANG, Q.; TANG, L. X. Heuristic Particle Swarm Optimization for Batching and Scheduling Flowshop with Sequence-Independent Setup Time. **Applied Mechanics and Materials**, v. 130–134, p. 2504–2507, 2011.

TASGETIREN, M. F.; KIZILAY, D.; PAN, Q.; SUGANTHAN, P. N. . Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. **Computers and Operation Research**, v. 77, p. 111–126, 2017. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2016.07.002>>. .

TASGETIREN, M. F.; PAN, Q. K.; KIZILAY, D.; SUER, G. A populated local search with differential evolution for blocking flowshop scheduling problem. **2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings**, p. 2789–2796, 2015. IEEE.

TASGETIREN, M. F.; PAN, Q. K.; SUGANTHAN, P. N.; CHEN, A. H. L. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. **Information Sciences**, v. 181, n. 16, p. 3459–3475, 2011. Elsevier Inc. Disponível em: <<http://dx.doi.org/10.1016/j.ins.2011.04.018>>. .

TASGETIREN, M. F.; PAN, Q.; KIZILAY, D.; GAO, K. A Variable Block Insertion Heuristic for the Blocking Flowshop Scheduling Problem with Total Flowtime Criterion. **Algorithms**, v. 9, n. 4, p. 1–24, 2016.

TASGETIREN, M. F.; SEVKLI, M.; LIANG, Y.; GENCYILMAZ, G. Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem. *Lecture Notes in Computer Science*. **Anais...** v. 3172, p.382–389, 2004. Berlin: Springer.

TIWARI, A.; CHANG, P.-C.; TIWARI, M.; KOLLANOOR, N. J. A pareto block-based estimation and distribution algorithm for multi-objective permutation flow shop scheduling. **International Journal of Production Economics**, v. 53, n. 3, p. 793–834, 2015.

TOUMI, S. .; JARBOUI, B. .; EDDALY, M. .; REBAÏ, A. . Branch-and-bound algorithm for solving blocking flowshop scheduling problems with makespan criterion. **International Journal of Mathematics in Operational Research**, v. 10, n. 1, p. 34–48, 2017.

TOUMI, S.; JARBOUI, B.; EDDALY, M.; REBAÏ, A. Solving Blocking Flowshop Scheduling Problem with Branch and Bound Algorithm. *Control, Decision and Information Technologies (CoDIT)*, 2013 International Conference on. IEEE. **Anais...** . p.746–749, 2013a.

TOUMI, S.; JARBOUI, B.; EDDALY, M.; REBAÏ, A. An efficient branch and bound algorithm to solve the permutation flowshop scheduling problem with blocking constraints. 2013 5th International Conference on Modeling, Simulation and Applied Optimization, ICMSAO 2013. **Anais...** . p.1–6, 2013b.

TOUMI, S.; JARBOUI, B.; EDDALY, M.; REBAÏ, A. Lower bounds on the total tardiness and total weighted tardiness for scheduling flowshop with blocking. 2013 International Conference on Advanced Logistics and Transport, ICALT 2013. **Anais...** . p.296–301, 2013c.

TOUMI, S.; JARBOUI, B.; EDDALY, M.; REBAÏ, A. Branch and Bound algorithm for solving blocking flowshop scheduling problem with total completion time. *Control, Decision and*

Information Technologies (CoDIT), 2013 International Conference on. IEEE. **Anais...** . p.746–749, 2013d.

TOUMI, S.; JARBOUI, B.; EDDALY, M.; REBAI, A. New lower bounds for the blocking flowshop scheduling problem to minimize the total completion time criterion. 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO). **Anais...** . p.1–5, 2013e. IEEE.

TRABELSI, W.; SAUVEY, C.; SAUER, N. **Complexity and mathematical model for flowshop problem subject to different types of blocking constraint**. IFAC, 2011.

TRABELSI, W.; SAUVEY, C.; SAUER, N. Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems. **Computers and Operations Research**, v. 39, n. 11, p. 2520–2527, 2012. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2011.12.022>>. .

TSANG, A. H. C.; JARDINE, A. K. S.; KOLODNY, H.; et al. Integrated production planning and preventive maintenance in deteriorating production systems. **Information Sciences**, v. 278, n. 1, p. 841–861, 2014. Elsevier Inc. Disponível em: <<http://dx.doi.org/10.1016/j.ins.2014.03.097>>. .

TSENG, L.-Y.; LIN, Y.-T. A hybrid genetic algorithm for no-wait flowshop scheduling problem. **International Journal of Production Economics**, v. 128, n. 1, p. 144–152, 2010. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-77957784429&partnerID=tZOtx3y1>>. Acesso em: 21/11/2014.

VAHEDI-NOURI, B.; FATTAHI, P.; RAMEZANIAN, R. Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. **International Journal of production Research**, v. 51, n. 12, p. 3501–3515, 2013.

VAHEDI-NOURI, B.; FATTAHI, P.; TAVAKKOLI-MOGHADDAM, R.; RAMEZANIAN, R. A general flow shop scheduling problem with consideration of position-based learning effect and multiple availability constraints. **International Journal of Advanced Manufacturing Technology**, v. 73, n. 5–8, p. 601–611, 2014.

VALLADA, E.; RUIZ, R. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem □. **Omega**, v. 38, n. 1–2, p. 57–67, 2010. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.omega.2009.04.002>>. .

VAN, L. E.; CUMMINGS, P. T.; VAN, D. LE. SCHEDULING OF SERIAL MULTIPRODUCT BATCH PROCESSES VIA SIMULATED ANNEALING. **Computers & Chemical Engineering**, v. 14, n. 12, p. 1351–1362, 1990.

VENTURA, J. A.; YOON, S.-H. A new genetic algorithm for lot-streaming flow shop scheduling with limited capacity buffers. **Journal of Intel**, v. 24, p. 1185–1196, 2013.

VO, N. V.; FOUILLET, P.; LENTÉ, C. Elaboration of General Lower Bounds for the Total Completion Time in Flowshop Scheduling Problems through MaxPlus Approach. In: E. Pinson; F. Valente; B. Vitoriano (Eds.); Operations Research and Enterprise Systems. **Anais...** . p.285–299, 2015. Cham: Springer International Publishing.

WANG, C.; SONG, S.; GUPTA, J. N. D.; WU, C. A three-phase algorithm for flowshop

scheduling with blocking to minimize makespan. **Computers and Operations Research**, v. 39, n. 11, p. 2880–2887, 2012. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2012.02.020>>. .

WANG, F.; DENG, G.; JIANG, T.; ZHANG, S. Multi-Objective Parallel Variable Neighborhood Search for Energy Consumption Scheduling in Blocking Flow Shops. **IEEE Access**, v. 6, p. 68686–68700, 2018. IEEE.

WANG, H.; WANG, W.; SUN, H.; et al. A modified cuckoo search algorithm for flow shop scheduling problem with blocking. 2015 IEEE Congress on Evolutionary Computation (CEC). **Anais...** . p.456–463, 2015. Sendai: IEEE. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7256925>>. .

WANG, L.; PAN, Q.-K.; FATIH TASGETIREN, M.; TASGETIREN, M. F. Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. **Expert Systems with Applications**, v. 37, n. 12, p. 7929–7936, 2010. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.eswa.2010.04.042>>. .

WANG, L.; PAN, Q.-K.; SUGANTHAN, P. N. N.; WANG, W.-H.; WANG, Y.-M. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. **Computers & Operations Research**, v. 37, n. 3, p. 509–520, 2010. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2008.12.004>>. .

WANG, L.; PAN, Q.-K.; TASGETIREN, M. F. A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. **Computers & Industrial Engineering**, v. 61, n. 1, p. 76–83, 2011.

WANG, L.; ZHANG, L.; ZHENG, D. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. **computers & oper**, v. 33, p. 2960–2971, 2006.

WANG, S.; LIU, M. A branch and bound algorithm for single-machine production scheduling integrated with preventive maintenance planning. **International Journal of Production Research**, v. 51, n. 3, p. 847–868, 2013. Disponível em: <<http://10.0.4.56/00207543.2012.676683%5Cnhttps://ezp.lib.unimelb.edu.au/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=83932644&site=eds-live&scope=site>>. .

WANG, S.; LIU, M. Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method. **International Journal of Production Research**, v. 52, n. 5, p. 1495–1508, 2014.

WANG, X.; TANG, L. A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking. **Applied Soft Computing Journal**, v. 12, n. 2, p. 652–662, 2012. Elsevier B.V. Disponível em: <<http://dx.doi.org/10.1016/j.asoc.2011.09.021>>. .

WENG, M. X. Scheduling flow shops with limited buffer spaces. Proceedings of the 2000 Winter Simulation Conference. **Anais...** . p.1359–1363, 2000. Orlando: IEEE.

WIDMER, M.; HERTZ, A. A new heuristic method for the flow shop sequencing problem. **European Journal of Operational Research**, v. 41, n. 2, p. 186–193, 1989.

WIE, S.; PINEDO, M. On Minimizing the Expected Makespan and Flow Time in Stochastic Flow Shops with Blocking. **Mathematics of Operations Research**, v. 11, n. 2, p. 336–342, 1986.

XU, D.; WAN, L.; LIU, A.; YANG, D. L. Single machine total completion time scheduling problem with workload-dependent maintenance duration. **Omega (United Kingdom)**, v. 52, p. 101–106, 2015. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.omega.2014.11.002>>. .

XU, D.; YIN, Y.; LI, H. Scheduling jobs under increasing linear machine maintenance time. **Journal of Scheduling**, v. 13, n. 4, p. 443–449, 2010.

YANG, S. J. Single-machine scheduling problems simultaneously with deterioration and learning effects under deteriorating multi-maintenance activities consideration. **Computers and Industrial Engineering**, v. 62, n. 1, p. 271–275, 2012. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2011.09.014>>. .

YANG, S. J. Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. **Applied Mathematical Modelling**, v. 37, n. 5, p. 2995–3005, 2013. Elsevier Inc. Disponível em: <<http://dx.doi.org/10.1016/j.apm.2012.07.029>>. .

YANG, S. J.; YANG, D. L. Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. **Omega**, v. 38, n. 6, p. 528–533, 2010. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.omega.2010.01.003>>. .

YANG, W.; LIAO, C.-J. Survey of scheduling research involving setup times. **International Journal of Systems Science**, v. 30, n. September, p. 143–155, 1999.

YANG, Z.; LIU, C. A hybrid multi-objective gray wolf optimization algorithm for a fuzzy blocking flow shop scheduling problem. **Advances in Mechanical Engineering**, v. 10, n. 3, p. 1–13, 2018.

YENISEY, M. M.; YAGMAHAN, B. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. **Omega**, v. 45, p. 119–135, 2014. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.omega.2013.07.004>>. .

YU, A. J.; SEIF, J. Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA. **Computers and Industrial Engineering**, v. 97, p. 26–40, 2016. Elsevier Ltd. Disponível em: <<http://dx.doi.org/10.1016/j.cie.2016.03.024>>. .

YUAN, J.; SOUKHAL, A.; CHEN, Y.; LU, L. A note on the complexity of flow shop scheduling with transportation constraints. **European Journal of Operational Research**, v. 178, n. 3, p. 918–925, 2007.

YUAN, K.; SAUER, N. Application of EM algorithm to flowshop scheduling problems with a special blocking. IESM'07. **Anais...**, 2007. Beijing.

ZANDIEH, M.; GHOLAMI, M. An immune algorithm for scheduling a hybrid flow shop sequence-dependent setup times and machines with random breakdowns. **International Journal of Production Research**, v. 47, n. 24, p. 6999–7027, 2009.

ZAROOK, Y.; REZAEIAN, J.; TAVAKKOLI-MOGHADDAM, R.; MAHDAVI, I.;

JAVADIAN, N. Minimization of makespan for the single batch-processing machine scheduling problem with considering aging effect and multi-maintenance activities. **International Journal of Advanced Manufacturing Technology**, v. 76, n. 9–12, p. 1879–1892, 2014.

ZHANG, C.; XIE, Z.; SHAO, X.; TIAN, G. An effective VNSSA algorithm for the blocking flowshop scheduling problem with makespan minimization. **International Conference on Advanced Mechatronic Systems, ICAMechS**, v. 2015-Octob, p. 86–89, 2015.

ZHAO, C. LI; TANG, H. YONG. Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan. **Applied Mathematical Modelling**, v. 34, n. 3, p. 837–841, 2010. Elsevier Inc. Disponível em: <<http://dx.doi.org/10.1016/j.apm.2009.07.002>>. .

ZHAO, F.; TANG, J.; WANG, J.; JONRINALDI, J. An improved particle swarm optimisation with a linearly decreasing disturbance term for flow shop scheduling with limited buffers. **International Journal of Computer Integrated Manufacturing**, v. 27, n. 5, p. 488–499, 2014.

ZHENG, Y. Water wave optimization : A new nature-inspired metaheuristic. **Computers and Operation Research**, v. 55, p. 1–11, 2015. Elsevier. Disponível em: <<http://dx.doi.org/10.1016/j.cor.2014.10.008>>. .

ZITZLER, E.; LAUMANN, M.; THIELE, L. SPREA2: Improving the strength pareto evolutionary algorithm. **TIK-Report**, v. 103, p. 12–19, 2001.

APÊNDICE A – DESVIOS RELATIVOS DA HEURÍSTICA HPF2 PARA OS CENÁRIOS YS E RGM

Tabela 55 - Desvios relativos para todas as combinações entre μ e λ da heurística HPF2 no cenário YS

μ/λ	0	0,05	0,1	0,15	0,2	0,25	0,3	0,35	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1
0	0,506	0,502	0,503	0,501	0,496	0,488	0,496	0,498	0,488	0,488	0,502	0,491	0,492	0,482	0,480	0,481	0,482	0,480	0,480	0,480	0,481
0,05	0,511	0,504	0,500	0,490	0,485	0,490	0,494	0,495	0,485	0,475	0,504	0,466	0,459	0,467	0,458	0,466	0,462	0,468	0,467	0,459	0,458
0,1	0,514	0,514	0,520	0,512	0,508	0,502	0,502	0,499	0,496	0,480	0,514	0,460	0,448	0,448	0,440	0,448	0,451	0,456	0,454	0,445	0,446
0,15	0,512	0,509	0,513	0,502	0,501	0,491	0,493	0,505	0,514	0,500	0,509	0,487	0,474	0,479	0,468	0,483	0,494	0,490	0,486	0,479	0,479
0,2	0,513	0,506	0,507	0,500	0,495	0,489	0,496	0,512	0,519	0,503	0,506	0,492	0,478	0,485	0,480	0,487	0,497	0,486	0,486	0,479	0,480
0,25	0,522	0,511	0,502	0,485	0,479	0,475	0,480	0,496	0,496	0,486	0,511	0,481	0,469	0,478	0,472	0,488	0,495	0,492	0,484	0,478	0,475
0,3	0,500	0,494	0,492	0,476	0,465	0,460	0,468	0,479	0,482	0,469	0,494	0,470	0,466	0,473	0,463	0,475	0,482	0,484	0,480	0,479	0,485
0,35	0,500	0,498	0,488	0,476	0,475	0,460	0,461	0,469	0,472	0,464	0,498	0,462	0,461	0,464	0,457	0,471	0,475	0,476	0,471	0,464	0,471
0,4	0,508	0,506	0,501	0,494	0,493	0,487	0,492	0,493	0,489	0,478	0,506	0,471	0,471	0,471	0,463	0,477	0,479	0,474	0,472	0,469	0,475
0,45	0,502	0,495	0,487	0,482	0,478	0,477	0,485	0,489	0,484	0,481	0,495	0,468	0,471	0,471	0,466	0,480	0,482	0,475	0,475	0,477	0,486
0,5	0,502	0,494	0,492	0,488	0,488	0,481	0,489	0,490	0,492	0,492	0,494	0,487	0,487	0,481	0,467	0,484	0,483	0,476	0,475	0,477	0,483
0,55	0,483	0,481	0,478	0,469	0,471	0,467	0,479	0,484	0,493	0,496	0,481	0,487	0,491	0,486	0,475	0,492	0,493	0,490	0,489	0,489	0,495
0,6	0,476	0,473	0,468	0,462	0,464	0,449	0,461	0,466	0,475	0,481	0,473	0,474	0,479	0,476	0,468	0,484	0,483	0,479	0,483	0,476	0,479
0,65	0,478	0,468	0,471	0,465	0,465	0,456	0,469	0,470	0,485	0,487	0,468	0,472	0,476	0,471	0,468	0,479	0,474	0,470	0,470	0,473	0,480
0,7	0,485	0,474	0,474	0,468	0,468	0,466	0,475	0,480	0,498	0,497	0,474	0,475	0,479	0,472	0,465	0,475	0,471	0,464	0,462	0,472	0,482
0,75	0,495	0,484	0,488	0,485	0,486	0,480	0,488	0,490	0,508	0,503	0,484	0,479	0,482	0,479	0,468	0,476	0,473	0,464	0,461	0,463	0,476
0,8	0,481	0,472	0,474	0,473	0,473	0,465	0,474	0,479	0,503	0,500	0,472	0,480	0,485	0,483	0,470	0,481	0,476	0,473	0,468	0,472	0,484
0,85	0,476	0,461	0,478	0,478	0,476	0,469	0,479	0,479	0,506	0,502	0,461	0,494	0,501	0,494	0,479	0,487	0,481	0,475	0,471	0,474	0,484
0,9	0,500	0,480	0,486	0,487	0,485	0,477	0,489	0,487	0,508	0,509	0,480	0,490	0,495	0,489	0,473	0,484	0,474	0,467	0,467	0,467	0,479
0,95	0,497	0,482	0,489	0,492	0,493	0,486	0,499	0,498	0,512	0,511	0,482	0,490	0,497	0,494	0,479	0,486	0,481	0,473	0,472	0,471	0,485
1	0,505	0,490	0,490	0,485	0,489	0,486	0,495	0,494	0,509	0,505	0,490	0,490	0,499	0,493	0,478	0,487	0,473	0,467	0,467	0,460	0,481

Fonte: Autor, 2019

Tabela 56 - Desvios relativos para todas as combinações entre μ e λ da heurística HPF2 no cenário RGM

μ/λ	0	0,05	0,1	0,15	0,2	0,25	0,3	0,35	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1
0	0,936	0,931	0,944	0,943	0,952	0,963	0,955	0,955	0,970	0,973	0,931	0,972	0,970	0,960	1,521	0,949	1,519	0,949	0,943	1,520	1,512
0,05	0,522	0,509	0,531	0,540	0,559	0,560	0,554	0,565	0,579	0,575	0,509	0,582	0,576	0,561	0,561	0,570	0,576	0,572	0,563	0,564	0,555
0,1	0,525	0,512	0,526	0,528	0,543	0,542	0,535	0,546	0,567	0,570	0,512	0,579	0,581	0,571	0,575	0,571	0,564	0,560	0,545	0,543	0,526
0,15	0,524	0,516	0,517	0,530	0,546	0,538	0,528	0,541	0,552	0,557	0,516	0,558	0,571	0,560	0,554	0,560	0,565	0,563	0,563	0,565	0,558
0,2	0,522	0,518	0,525	0,531	0,544	0,540	0,541	0,551	0,574	0,571	0,518	0,570	0,573	0,567	0,574	0,580	0,586	0,574	0,581	0,587	0,575
0,25	0,534	0,531	0,529	0,523	0,531	0,528	0,524	0,533	0,546	0,546	0,531	0,547	0,552	0,562	0,566	0,572	0,581	0,573	0,574	0,570	0,568
0,3	0,513	0,517	0,517	0,512	0,510	0,507	0,505	0,512	0,510	0,508	0,517	0,522	0,529	0,540	0,546	0,554	0,555	0,550	0,553	0,551	0,551
0,35	0,537	0,549	0,555	0,549	0,542	0,537	0,541	0,548	0,549	0,543	0,549	0,561	0,563	0,569	0,572	0,564	0,558	0,555	0,557	0,556	0,551
0,4	0,521	0,519	0,523	0,520	0,525	0,525	0,527	0,539	0,543	0,540	0,519	0,557	0,559	0,568	0,571	0,572	0,574	0,583	0,576	0,577	0,572
0,45	0,537	0,531	0,530	0,527	0,526	0,516	0,519	0,534	0,538	0,531	0,531	0,544	0,554	0,556	0,560	0,568	0,563	0,569	0,564	0,571	0,570
0,5	0,539	0,540	0,537	0,538	0,536	0,532	0,529	0,536	0,537	0,533	0,540	0,537	0,541	0,550	0,566	0,570	0,568	0,570	0,569	0,571	0,566
0,55	0,535	0,541	0,544	0,543	0,534	0,531	0,522	0,523	0,523	0,524	0,541	0,522	0,526	0,530	0,541	0,550	0,545	0,543	0,543	0,557	0,553
0,6	0,546	0,562	0,568	0,565	0,553	0,553	0,553	0,549	0,551	0,542	0,562	0,542	0,550	0,552	0,570	0,575	0,566	0,561	0,559	0,559	0,553
0,65	0,531	0,547	0,543	0,536	0,524	0,526	0,529	0,530	0,536	0,528	0,547	0,533	0,545	0,550	0,560	0,568	0,569	0,569	0,565	0,572	0,564
0,7	0,539	0,547	0,555	0,559	0,540	0,536	0,536	0,537	0,545	0,536	0,547	0,532	0,544	0,551	0,562	0,571	0,569	0,564	0,568	0,577	0,560
0,75	0,533	0,541	0,551	0,555	0,546	0,544	0,547	0,550	0,551	0,540	0,541	0,530	0,536	0,540	0,548	0,549	0,548	0,540	0,534	0,543	0,532
0,8	0,539	0,546	0,557	0,559	0,540	0,552	0,557	0,558	0,548	0,540	0,546	0,532	0,544	0,546	0,551	0,557	0,557	0,550	0,539	0,546	0,535
0,85	0,538	0,545	0,549	0,550	0,531	0,536	0,538	0,541	0,527	0,523	0,545	0,515	0,530	0,538	0,547	0,553	0,547	0,538	0,538	0,544	0,535
0,9	0,519	0,531	0,534	0,538	0,520	0,530	0,533	0,537	0,527	0,519	0,531	0,509	0,518	0,527	0,534	0,541	0,540	0,534	0,535	0,535	0,526
0,95	0,675	0,689	0,690	0,691	0,684	0,691	0,687	0,682	0,675	0,662	0,689	0,660	0,666	0,641	0,673	0,680	0,670	0,666	0,663	0,661	0,651
1	0,519	0,531	0,535	0,537	0,535	0,538	0,532	0,536	0,532	0,530	0,531	0,534	0,536	0,536	0,541	0,542	0,540	0,540	0,539	0,540	0,532

Fonte: Autor, 2019